Habilitation Thesis

Dr. Andreas Emil Feldmann

Parameterized Approximation Algorithms in Network Design and Clustering

Department of Applied Mathematics (KAM)

Prague 2020
Preface

This habilitation thesis gives an overview of some recent results obtained by the author together with various collaborators in the area of parameterized approximation algorithms. This research area in the intersection of fixed-parameter tractability and approximation algorithms has gained growing attention in recent years. While a large variety of algorithmic topics have seen advancements using the paradigm of parameterized approximations (for an overview see the recent survey in [9]), the author’s contributions are mostly concentrated on the topics of network design and clustering. Accordingly, this thesis presents the author’s results on these two topics in separate sections. Each of these two sections gives a brief overview of the obtained results, after which the used techniques are presented in more detail.

The results appear in several papers listed below, which were published between 2015 and 2020. The papers are also attached to the appendix of this thesis. Some of the attached papers are extended abstracts of conference proceedings, and thus do not contain the full details due to strict page limitations. Therefore a link to a full version of each paper is provided in the list below.

Please note that several passages of this thesis are taken verbatim (with some modifications) from the author’s publications. In particular, Section 1 uses parts of [9], Sections 2.1 and 2.2 contain excerpts from [3; 9], Section 2.3 includes some of [10], Section 3.1 contains parts of [5; 8; 9; 11], Section 3.2 lends from [4; 12], and Section 3.3 uses content from [7; 11].


Contents

Parameterized Approximation Algorithms in Network Design and Clustering 5
1 Parameterized approximation ............................................. 5
  1.1 Preliminaries ..................................................... 6
2 Network design .......................................................... 8
  2.1 Undirected graphs ................................................... 9
  2.2 Directed Graphs .................................................... 10
  2.3 Computing exact solutions in directed graphs ................. 14
3 Clustering ................................................................. 16
  3.1 Metrics modelling transportation networks ..................... 17
  3.2 The k-MEDIAN problem and its variants ....................... 19
    3.2.1 Low doubling metrics .................................... 19
    3.2.2 Low highway dimension graphs ......................... 21
    3.2.3 Metric embeddings ....................................... 23
  3.3 The k-CENTER problem ........................................... 26
References to related work ................................................. 30

Appendix: Papers Summarized in This Thesis 35
[1] FPT Inapproximability of Directed Cut and Connectivity Problems ...... 36
[2] Tight Bounds for Planar Strongly Connected Steiner Subgraph with Fixed Number of Terminals (and Extensions) .................. 56
[3] Parameterized Approximation Algorithms for Bidirected Steiner Network Problems ......................................................... 103
[5] Travelling on Graphs with Small Highway Dimension ............... 139
[6] Parameterized Approximation Schemes for Steiner Trees with Small Number of Steiner Vertices ...................................... 154
[8] A (1 + ε)-Embedding of Low Highway Dimension Graphs into Bounded Treewidth .......................................................... 182
[12] Polynomial Time Approximation Schemes for Clustering in Low Highway Dimension Graphs .................................................. 311
Parameterized Approximation Algorithms in Network Design and Clustering

1 Parameterized approximation

In their seminal papers of the mid 1960s, Cobham [Cob64] and Edmonds [Edm65] independently phrased what is now known as the Cobham-Edmonds thesis. It states that an optimization problem is feasibly solvable if it admits an algorithm with the following two properties:

1. **Accuracy:** the algorithm should always compute the best possible (optimum) solution.
2. **Efficiency:** the runtime of the algorithm should be polynomial in the input size $n$.

Shortly after the Cobham-Edmonds thesis was formulated, the development of the theory of NP-hardness and reducibility identified a whole plethora of problems that are seemingly intractable, i.e., for which algorithms with the above two properties do not seem to exist. Even though the reasons for this phenomenon remain elusive up to this day, this has not hindered the development of algorithms for such problems. To obtain an algorithm for an NP-hard problem, at least one of the two properties demanded by the Cobham-Edmonds thesis needs to be relaxed. Ideally, the properties are relaxed as little as possible, in order to stay close to the notion of feasible solvability suggested by the thesis.

A very common approach is to relax the accuracy condition, which means aiming for approximation algorithms [Vaz01; WS11]. The idea here is to use only polynomial time to compute an $\alpha$-approximation, i.e., a solution that is at most a factor $\alpha$ times worse than the optimum solution obtainable for the given input instance. Such an algorithm may also be randomized, i.e., there is either a high probability that the output is an $\alpha$-approximation, or the runtime is polynomial in expectation.

In a different direction, several relaxations of the efficiency condition have also been proposed. Popular among these is the notion of parameterized algorithms [Cyg+15; DF13]. Here the input comes together with some parameter $k \in \mathbb{N}$, which describes some property of the input and can be expected to be small in typical applications. The idea is to isolate the seemingly necessary exponential runtime of NP-hard problems to the parameter, while the runtime dependence on the input size $n$ remains polynomial. In particular, the algorithm should compute the optimum solution in $f(k)n^{O(1)}$ time, for some computable function $f : \mathbb{N} \to \mathbb{N}$ independent of the input size $n$. If such an algorithm exists for a problem it is fixed-parameter tractable (FPT), and the algorithm is correspondingly referred to as an FPT algorithm. Again, such an algorithm may be randomized.

Approximation and FPT algorithms have been studied extensively for the past few decades, and this has lead to a rich literature on algorithmic techniques and deep links to other research.
fields within mathematics. In this process the limitations of these approaches have also become apparent. Some NP-hard problems can fairly be considered to be feasibly solvable in the respective regimes, as they admit polynomial-time algorithms with small approximation factors, or can be shown to be solvable optimally with only a limited exponential runtime overhead due to the parameter. But many problems can also be shown not to admit any reasonable algorithms in either of these regimes, assuming some standard complexity assumptions. Thus considering only approximation and FPT algorithms, as has been mostly done in the past, we are seemingly stuck in a swamp of problems for which we have mathematical evidence that they cannot be feasibly solved.

To find a way out of this dilemma, an obvious possibility is to lift both the accuracy and the efficiency requirements of the Cobham-Edmonds thesis. In this way we obtain a parameterized $\alpha$-approximation algorithm, which computes an $\alpha$-approximation in $f(k)n^{O(1)}$ time for some computable function $f$, given an input of size $n$ with parameter $k$. The study of such algorithms had been suggested dating back to the early days of parameterized complexity (cf. [CC97; DF13; FG06]), and we refer the readers to a survey of Marx [Mar08] for discussions on earlier results in the area, and our survey in [9] for an overview of more recent developments.

The aim of this thesis is to present the contributions of the author to the field of parameterized approximation algorithms. The main focus of the author’s work has been on problems arising in network design and clustering. These are well-studied areas for both parameterized and approximation algorithms and therefore constitute natural starting points to develop a theory of parameterized approximation algorithms. In Section 2 we present the results in network design, while Section 3 gives an overview of those for clustering. Each section begins with a brief overview of the obtained results by the author and several collaborators, after which a more detailed account of the used techniques to obtain these results is given. Before this however, in Section 1.1 we give a more formal introduction of several concepts used in the field.

1.1 Preliminaries

In this section, we review several notions relevant to the study of parameterized approximations, and how they relate to previously studied concepts in the more classic fields of parameterized and approximation algorithms. We will not review common graph theoretic notions such as planarity or treewidth in this introduction, and instead refer to the literature [Cyg+15; Die12] and also later chapters in the appendix of this thesis, which contain some such definitions.

Parameterized approximation algorithms. As already defined above, an FPT algorithm computes the optimum solution in $f(k)n^{O(1)}$ time for some parameter $k$ and computable function $f : \mathbb{N} \to \mathbb{N}$ on inputs of size $n$. An algorithm that computes the optimum solution in $f(k)n^{g(k)}$ time for some parameter $k$ and computable functions $f, g : \mathbb{N} \to \mathbb{N}$, is called a slice-wise polynomial (XP) algorithm. If the parameter is the approximation factor, i.e., the algorithm computes a $(1 + \varepsilon)$-approximation in $f(\varepsilon)n^{g(\varepsilon)}$ time, then it is called a polynomial-time approximation scheme (PTAS). The latter type of algorithm has been studied avant la lettre for quite some time, where it is assumed that $\varepsilon$ is a constant and thus the runtime is polynomial. Also the corresponding FPT algorithm has been studied before, which computes a $(1 + \varepsilon)$-approximation in $f(\varepsilon)n^{O(1)}$ time, and is referred to as an efficient polynomial-time approximation scheme (EPTAS). While it may be unusual to view these algorithms from the perspective of parametrizations, we will specifically do so in this thesis in order to obtain a more nuanced view of the complexity of the studied problems (especially in Section 3.2).

As also mentioned above, a parameterized $\alpha$-approximation algorithm computes an $\alpha$-approximation in $f(k)n^{O(1)}$ time for some parameter $k$ on inputs of size $n$. If $\alpha$ can be set to $1 + \varepsilon$
for any $\epsilon > 0$ and the runtime is $f(k, \epsilon) n^{g(\epsilon)}$, then we obtain a parameterized approximation scheme (PAS) for parameter $k$. Note that this runtime is only truly FPT if we assume that $\epsilon$ is constant, and a PAS is thus the corresponding notion to a PTAS. If we forbid this and consider $\epsilon$ as a parameter as well, i.e., the runtime should be of the form $f(k, \epsilon) n^{O(1)}$, then we obtain an efficient parameterized approximation scheme (EPAS), which is the corresponding notion to an EPTAS.

**Kernelization.** A topic closely related to FPT algorithms is kernelization. Here the idea is that an instance is efficiently pre-processed by removing the “easy parts” so that only the (NP-)hard core of the instance remains. More concretely, a kernelization algorithm takes an instance $I$ and a parameter $k$ of some problem and computes a new instance $I'$ with parameter $k'$ of the same problem. The runtime of this algorithm is polynomial in the size of the input instance $I$ and $k$, while the size of the output $I'$ and $k'$ is bounded as a function of the input parameter $k$. For optimization problems it should also be the case that any optimum solution to $I'$ can be converted to an optimum solution of $I$ in polynomial time. The new instance $I'$ is called the kernel of $I$ (for parameter $k$). A fundamental result in fixed-parameter tractability is that an (optimization) problem parameterized by $k$ is FPT if and only if it admits a kernelization algorithm for the same parameter [Cyg+15]. However the size of the guaranteed kernel will in general be exponential (or worse) in the input parameter. Therefore an interesting question is whether an NP-hard problem admits small kernels of polynomial size. This can be interpreted as meaning that the problem has a very efficient pre-processing algorithm, which can be used to compress the instance prior to solving the kernel. This also provides an additional dimension to the parameterized complexity landscape, and kernelization has therefore been developed into a research area in its own right.

Kernelization has played a fundamental role in the development of FPT algorithms, where often a pre-processing step is used to simplify the structure of the input instance. It is therefore only natural to consider such pre-processing algorithms for parameterized approximation algorithms as well. The notion we will be concerned with here was introduced by Lokshtanov et al. [Lok+17]. They define an $\alpha$-approximate kernelization algorithm, which computes a kernel $I'$ such that any $\beta$-approximation for $I'$ can be converted into an $\alpha\beta$-approximation to the input instance $I$ in polynomial time. Again the size of $I'$ and $k'$ need to be bounded as a function of the input parameter $k$, and the algorithm needs to run in polynomial time. The instance $I'$ is now called an $\alpha$-approximate kernel. Analogous to exact kernels, any problem has a parameterized $\alpha$-approximation algorithm if and only if it admits an $\alpha$-approximate kernel for the same parameter [Lok+17], which however might be of exponential size in the parameter. As before, studying the existence of polynomial-sized approximate kernels adds an additional dimension to the complexity landscape of parameterized approximation algorithms, and approximate kernels are hence interesting to study in their own right.

An $\alpha$-approximate kernelization algorithm that computes a polynomial-sized kernel, and for which we may set $\alpha$ to $1 + \epsilon$ for any $\epsilon > 0$, is called a polynomial-sized approximate kernelization scheme (PSAKS). In this case $\epsilon$ is necessarily considered to be a constant, since any kernelization algorithms needs to run in polynomial time.

Kernelization (and FPT) algorithms often come with a set of reduction rules, which roughly speaking constitute steps to simplify the input and are applied repeatedly until some core instance (typically a kernel) is left. Formally, a reduction rule is a polynomial time algorithm, which takes an instance $I$ and a parameter $k$ as input, and outputs a new instance $I'$ and parameter $k'$ (but in contrast to a kernelization algorithm, the size of the new instance is not necessarily bounded after only one application of a reduction rule). Furthermore, in the context of approximate kernels, a reduction rule is said to be strictly $\alpha$-safe if there exists a polynomial
time algorithm to convert a $\beta$-approximation to $I'$ into a $\max\{\alpha, \beta\}$-approximation to $I$.\footnote{The strictness refers to the fact that the approximation factor is bounded by $\max\{\alpha, \beta\}$ instead of $\alpha \beta$. This is needed in order to apply reduction rules repeatedly without losing the guaranteed approximation factor. See [Lok+17] for more details.} A set of reduction rules is then applied repeatedly until some desired property of the resulting instance is met (for kernelizations this property typically is that the size of the final instance is bounded as a function of the input parameter $k$).

**Complexity-theoretic assumptions.** We assume that the reader is familiar with common complexity classes used to prove algorithmic lower bounds for parameterizations, approximations, and kernelizations, such as P, NP, W[\textit{t}], APX, and coNP/poly. For sake of brevity we will not define these classes here and instead refer to the literature [Cyg+15; FG06; WS11].

The Exponential Time Hypothesis (ETH) is often used to obtain concrete runtime lower bounds for parameterized problems. It assumes that 3SAT cannot be solved in $2^{o(n)}$ time, where $n$ is the number of variables. A less known assumption is the Gap Exponential Time Hypothesis (Gap-ETH), which is stronger than ETH and useful to obtain lower bounds for parameterized approximations. It assumes that there exists some constant $\delta > 0$ such that there is no $2^{o(n)}$ time algorithm to decide whether all or at most a $(1 - \delta)$-fraction of the clauses of a given 3SAT formula are satisfiable. Here the assumed algorithms may be deterministic or randomized (the latter being the stronger assumption).

## 2 Network design

In network design the task is to connect some set of vertices in an edge-weighted graph in the cheapest possible way. To give an example, a prominent problem of this type is the Steiner Tree problem. Here a subset of the vertices (called terminals) is given as part of the input, and the objective is to connect all terminals by a tree of minimum weight in the graph. This fundamental problem and its variants have been widely studied in the past, both on undirected and directed input graphs.

In Section 2.1 we focus on undirected input graphs. Our main results here are a PAS and a PSAKS for Steiner Tree parameterized by the number of non-terminals (Steiner vertices) contained in the optimum solution. In Section 2.2 we turn to directed graphs, where we first discuss the Directed Steiner Tree problem for the same parameter. In summary, we show that a PAS only exists in the unweighted case, but a PSAKS does not exist even then. Next we consider the more standard parameter given by the number of terminals, for which a different directed variant of Steiner Tree called Strongly Connected Steiner Subgraph is known to have a parameterized 2-approximation. We give a lower bound showing that this is best possible. For the more general Directed Steiner Network problem we prove that a PAS and a PSAKS exist on planar directed graphs that are also bidirected, which means that for every edge the reverse edge exists as well and has the same weight. We then present several hardness results showing that our PAS and PSAKS for this problem on planar bidirected graphs are in a sense best possible.

Finally, in Section 2.3 we present a dichotomy result on computing exact solutions for Directed Steiner Network parameterized by the number of terminals. We summarize several interesting consequences of the algorithm due to this result, including its application to derive the above-mentioned PAS for planar bidirected graphs.
2.1 Undirected graphs

A well-studied parameter for Steiner Tree is the number of terminals, for which the problem has been known to be FPT since the early 1970s due to the work of Dreyfus and Wagner [DW71]. Their algorithm is based on dynamic programming and runs in $3^k n^{O(1)}$ time if $k$ is the number of terminals. Faster algorithms based on the same ideas with runtime $(2 + \delta)^k n^{O(1)}$ for any constant $\delta > 0$ exist [Fuc+07] (here the degree of the polynomial depends on $\delta$). The unweighted Steiner Tree problem also admits a $2^k n^{O(1)}$ time algorithm [Ned09] using a different technique based on subset convolution. On the other hand, no exact polynomial-sized kernel exists [DLS14] for the Steiner Tree problem, unless NP $\subseteq$ coNP/poly. Interestingly though, a PSAKS can be obtained [Lok+17].

This approximate kernel is based on a well-known fact proved by Borchers and Du [BD97], which is very useful to obtain approximation algorithms for the Steiner Tree problem, and on which several of our results are based as well. On a high level, it states that any Steiner tree can be covered by smaller trees containing few terminals, such that these trees do not overlap much. More formally, a full-component is a subtree of a Steiner tree, for which the leaves coincide with its terminals. For the optimum Steiner tree $T$ and any $\varepsilon > 0$, there exist full-components $C_1, \ldots, C_\ell$ of $T$ such that

1. each full-component $C_i$ contains at most $2^{\lceil 1/\varepsilon \rceil}$ terminals (leaves),
2. the sum of the weights of the full-components is at most $1 + \varepsilon$ times the cost of $T$, and
3. taking any collection of Steiner trees $T_1, \ldots, T_\ell$, such that each tree $T_i$ connects the subset of terminals that forms the leaves of full-component $C_i$, the union $\bigcup_{i=1}^{\ell} T_i$ is a feasible solution to the input instance.

Not knowing the optimum Steiner tree, it is not possible to know the subsets of terminals of the full-components corresponding to the optimum. However, it is possible to compute the optimum Steiner tree for every subset of terminals of size at most $2^{\lceil 1/\varepsilon \rceil}$ using an FPT algorithm for Steiner Tree. The time to compute all these solutions is $k^{O(2^{1/\varepsilon})} n^{O(1)}$, using for instance the Dreyfus and Wagner [DW71] algorithm. Now the above three properties guarantee that the graph given by the union of all the computed Steiner trees, contains a $(1 + \varepsilon)$-approximation for the input instance. In fact, the best polynomial time approximation algorithm known to date [Byr+13] uses an iterative rounding procedure to find a $\ln(4)$-approximation of the optimum solution in the union of these Steiner trees. To obtain a kernel, the union needs to be sparsified, since it may contain many Steiner vertices and also the edge weights might be very large. Lokshtanov et al. [Lok+17] show that the number of Steiner vertices can be reduced using standard techniques, while the edge weights can be encoded so that their space requirement is bounded in the parameter and the cost of any solution is distorted by at most a $1 + \varepsilon$ factor. The resulting graph is thus a PSAKS parameterized by the number $k$ of terminals.

A natural alternative parameter to the number of terminals is to consider the vertices remaining in the optimum tree after removing the terminals: a folklore result states that Steiner Tree is W[2]-hard parameterized by the number of non-terminals (called Steiner vertices) in the optimum solution. At the same time, unless P=NP there is no PTAS for the problem, as it is APX-hard [CC08]. However, we were able to show that both an EPAS and a PSAKS are obtainable when parametrizing by the number of Steiner vertices $p$ in the optimum.

To obtain both of these results, in [6] we devise a reduction rule that is based on the following observation: if the optimum tree contains few Steiner vertices but many terminals, then the tree must contain (1) a large component containing only terminals, or (2) a Steiner vertex that has many terminal neighbours. Intuitively, in case (2) we would like to add a large star to the
solution, such that the star has terminal leaves and small cost in the current graph. In case (1) we would like to add a cheap edge between two terminals. Note that such a single edge also is a star with terminal leaves. The reduction rule will therefore find the star with minimum weight per contained terminal and contract it, which can be done in polynomial time. This rule is applied until the number of terminals, which decreases after each use, falls below a threshold depending on the input parameter \( p \) and the desired approximation ratio \( 1 + \varepsilon \). Once the number of terminals is bounded by a function of \( p \) and \( \varepsilon \), the Dreyfus and Wagner [DW71] algorithm can be applied on the remaining instance, or a kernel can be computed using the PSAKS of Lokshtanov et al. [Lok+17]. We prove that our reduction rule does not distort solutions by much as long as the threshold is large enough, which results in the following theorem.

**Theorem 2.1** ([6]). For the Steiner Tree problem a \((1 + \varepsilon)\)-approximation can be computed in \(2^{O(p^2/\varepsilon^4)}n^{O(1)}\) time for any \( \varepsilon > 0 \), where \( p \) is the number of non-terminals in the optimum solution. Moreover, a \((1+\varepsilon)\)-approximate kernel of size \((p/\varepsilon)^{2^{O(1/\varepsilon)}}\) can be computed in polynomial time.

A natural question is whether this theorem is generalizable to other variants of Steiner Tree in undirected graphs, for instance the Steiner Forest problem, where a list of terminal pairs is given and the task is to find a minimum weight forest in the input graph connecting each pair. Parameterized by the number of terminals \( k \) it is not hard to show that Steiner Forest is FPT, since we may guess a partition of the terminals such that each set of the partition is contained in the same connected component of the optimum Steiner Forest solution. Since each connected component forms a tree, an FPT algorithm for Steiner Tree can then be used to compute a solution for each terminal set separately, which leads to a runtime of \( k^{O(k)}n^{O(1)} \). Also a PAS can be obtained for this parameter, using the same techniques as in [Lok+17] for Steiner Tree (cf. [6]).

If however the parameter is the number \( p \) of Steiner vertices in the optimum solution, then neither a PAS nor a PSAKS exists unless P=NP. This can be easily seen, since any Steiner vertex \( v \) of a Steiner Forest instance can be promoted to a trivial terminal pair that both equal \( v \). Now any solution to the new instance corresponds to a solution in the original instance of the same weight, and vice versa. As the new instance contains no Steiner vertices, a PAS or PSAKS for parameter \( p \) would imply a PTAS for Steiner Forest. However, the problem is APX-hard [CC08], and thus a PTAS would imply P=NP.

Nonetheless, we showed in [6] that using the same techniques as for Steiner Tree, it is possible to generalize Theorem 2.1 to the Steiner Forest problem, if the parameter \( p \) is combined with the number \( c \) of connected components of the optimum solution. This yields a PAS with runtime \(2^{O((p+c)/\varepsilon)}n^{O(1)}\) and a PSAKS of size \(((p+c)/\varepsilon)^{2^{O(1/\varepsilon)}}\).

It is also natural to ask whether Theorem 2.1 is generalizable to variants of Steiner Tree in directed graphs, which we turn to next.

### 2.2 Directed Graphs

The Directed Steiner Tree problem takes as input a terminal set with a special terminal called the root in a directed edge-weighted graph. The task is to compute a directed tree of minimum weight that contains a path from each terminal to the root. For the parameterization by the number \( k \) of terminals, the Directed Steiner Tree problem is FPT, using the same algorithms as for the undirected version [DW71; Fuc+07; Ned09]. In contrast to the undirected case however, this problem is much harder to approximate. It was shown [HK03] that no \( O(\log^{2-\varepsilon} k) \)-approximation can be computed in polynomial time, unless NP-hard problems can be solved in expected quasi-polynomial time. Moreover, for the parameterization by the number \( p \)
of Steiner vertices in the optimum solution we proved in [6] that no reasonable approximation can be computed in FPT time. This can be shown using a simple reduction from the DOMINATING SET problem via a hardness result in [CL16], resulting in the following formal statement.

**Theorem 2.2 ([6]).** The Directed Steiner Tree problem has no $g(p)$-approximation algorithm with runtime $f(p)n^{O(1)}$, for any computable functions $f$ and $g$, where $p$ is the number of Steiner vertices in the optimum solution, unless $W[1]=FPT$.

A notable special case is the unweighted Directed Steiner Tree problem. Here we showed [6] that a PAS is again obtainable using the number $p$ of Steiner vertices in the optimum as a parameter. Similar as for the undirected case, the algorithm uses a reduction rule that contracts components containing many terminals (depending on $p$ and $\varepsilon$), and then uses an FPT algorithm parameterized by the number of terminal to solve the remaining instance. One caveat however is that contractions in directed graphs are tricky, as they may introduce new paths that were non-existent before. For Directed Steiner Tree this issue can be circumvented by making sure that the contracted component always contains the root. For this however, a path from the root to the contracted terminals needs to be included in the contracted component, which may contain many Steiner vertices. However in the unweighted case, the number of vertices of a path directly translates to the weight of the path. This effectively implies that the reduction rule does not distort solutions by much, and thus we obtain a PAS, as stated formally below.

A natural question then becomes whether, as in the undirected case, a PSAKS exists for the unweighted Directed Steiner Tree problem. Recall that the PSAKS for Steiner Tree relies on the result by Borchers and Du [BD97], which decomposes an undirected solution into full-components with a small number of terminals each, such that the full-components overlap very little. It is not hard to see however, that this cannot work in directed graphs. More generally, we prove [6] that in contrast to the undirected case, no polynomial-sized $(2-\varepsilon)$-approximate kernelization exists for unweighted Directed Steiner Tree, unless $\text{NP} \subseteq \text{coNP/poly}$. It remains an intriguing question whether a polynomial-sized 2-approximate kernel exists.

**Theorem 2.3 ([6]).** For the unweighted Directed Steiner Tree problem a $(1+\varepsilon)$-approximation can be computed in $2^p/\varepsilon n^{O(1)}$ time for any $\varepsilon > 0$, where $p$ is the number of non-terminals in the optimum solution. However, no polynomial-sized $(2-\varepsilon)$-approximate kernelization exists, unless $\text{NP} \subseteq \text{coNP/poly}$.

We now turn back to the well-studied parameterization by the number $k$ of terminals, and consider other directed variants of Steiner Tree. One example is the STRONGLY CONNECTED STEINER SUBGRAPH problem, where a terminal set needs to be strongly connected in the cheapest possible way. In contrast to Directed Steiner Tree this problem is $W[1]$-hard parameterized by the number of terminals [GNS11], and again no $O((\log^{2-\varepsilon} n)$)-approximation can be computed in polynomial time [HK03], unless $\text{NP} \subseteq \text{ZTIME}(n^{polylog(n)})$. However, a 2-approximation can be computed in FPT time [CHK13] using the parameter $k$.

Interestingly, in [3] we showed that no improvement over this 2-approximation is possible when parameterizing by $k$. To obtain this hardness result we modified the reduction of Guo et al. [GNS11], who showed $W[1]$-hardness of the problem. In particular, this reduction was from the $W[1]$-hard CLIQUE problem. As a starting point we instead use the approximation variant of CLIQUE, namely the DENSEST $k$-SUBGRAPH problem, and a recent inapproximability result for the latter [DM18]. Additionally, we introduce appropriate edge weights in the reduction of Guo et al. [GNS11]. Together with the positive result of Chitnis et al. [CHK13] we obtain the following theorem, which to date is the only known tight parameterized approximation result for a problem that can be approximated better in FPT time than in polynomial time, but where the parameterized algorithm is not an approximation scheme.
Theorem 2.4 ([3; CHK13]). For the Strongly Connected Steiner Subgraph problem a 2-approximation can be computed in \((2 + \delta)^k n^{O(1)}\) time for any constant \(\delta > 0\), where \(k\) is the number of terminals. Moreover, under Gap-ETH no \((2 - \epsilon)\)-approximation can be computed in \(f(k)n^{O(1)}\) time for any \(\epsilon > 0\) and computable function \(f\).

A generalization of both Directed Steiner Tree and Strongly Connected Steiner Subgraph is the Directed Steiner Network\(^2\) problem, for which an edge-weighted directed graph is given together with a list of ordered terminal pairs. The aim is to compute the cheapest subgraph that contains a path from \(s\) to \(t\) for every terminal pair \((s, t)\). If \(k\) is the number of terminals, then for this problem no \(k^{1/4 - o(1)}\)-approximation can be computed in \(f(k)n^{O(1)}\) time [DM18] for any computable function \(f\), under Gap-ETH. But we showed in [3] that both a PAS and a PSAKS exist for the special case when the input graph is planar\(^3\) and bidirected, i.e., for every directed edge \(uv\) the reverse edge \(vu\) exists and has the same cost.

To obtain these two algorithms, in [3] we generalize the theorem of Borchers and Du [BD97] for Steiner Tree to the Directed Steiner Network problem on planar bidirected graphs. That is, we show that a planar optimum solution in a bidirected graph can be covered by planar graphs with at most \(2^{O(1/\epsilon)}\) terminals each, such that the sum of their costs is at most \(1 + \epsilon\) times the cost of the solution. Similar to how Borchers and Du [BD97] exploit the tree structure of a solution to Steiner Tree, in our case the planarity is exploited to make sure that the covering graphs contain few terminals while at the same time do not cost much more than the optimum. However, to make sure that the union of these covering graphs constitutes a feasible solution, we may need to add edges that are reverse to those in the solution, but are themselves not part of the solution. For this the underlying graph needs to be bidirected.

To formally state our contribution, we encode the demands of a Directed Steiner Network instance using a pattern graph \(H\): the vertex set of \(H\) is the terminal set of the input graph \(G\), and \(H\) contains the directed edge \(st\) if and only if \((s, t)\) is a demand. Hence the Directed Steiner Network problem asks for a minimum cost network \(N \subseteq G\) having an \(s \rightarrow t\) path for each edge \(st\) of \(H\). In the following, \(\text{cost}(N)\) denotes the cost of a graph (solution) \(N\), i.e., the sum of its edge weights.

Theorem 2.5 ([3]). Let \(G\) be a bidirected graph, and \(H\) a pattern graph on the terminal set \(R\) of \(G\). Let \(N \subseteq G\) be the cheapest planar solution to pattern \(H\). For any \(\epsilon > 0\), there exists a set of patterns \(\mathcal{H}\) such that

1. \(V(H') \subseteq R\) with \(|V(H')| \leq 2^{1+[1/\epsilon]}\) for each \(H' \in \mathcal{H}\),
2. given any feasible solutions \(N_{H'} \subseteq G\) for all \(H' \in \mathcal{H}\), the union \(\bigcup_{H' \in \mathcal{H}} N_{H'}\) of these solutions forms a feasible solution to \(H\), and
3. there exist feasible planar solutions \(N^*_{H'} \subseteq G\) for all \(H' \in \mathcal{H}\) such that \(\sum_{H' \in \mathcal{H}} \text{cost}(N^*_{H'}) \leq (1 + \epsilon) \cdot \text{cost}(N)\).

Analogous to Steiner Tree, we now compute solutions for every possible list of ordered pairs (i.e., pattern graphs) of at most \(2^{1+[1/\epsilon]}\) terminals. In contrast to Steiner Tree however, it is unlikely that Directed Steiner Network on planar bidirected graphs is FPT parameterized by the number of terminals (see below). Instead we use an XP algorithm with runtime \(2^{O(k^{3/2} \log k)}n^{O(\sqrt{k})}\), which can be obtained by exploiting our insights on computing optimum solutions to the Directed Steiner Network problem presented in [10] (cf. Section 2.3).

\(^2\)Sometimes also called Directed Steiner Forest; note however that the optimum is not necessarily a forest.

\(^3\)A directed graph is planar if its underlying undirected graph is.
Since each considered pattern graph has at most $2^{1+[1/\varepsilon]}$ terminals, the time needed to compute solutions for all of them can be bounded by $n^{2^{O(1/\varepsilon)}}$, which is polynomial if $\varepsilon$ is constant.

To obtain a PSAKS, after taking the union of all computed solutions, the number of Steiner vertices and the encoding length of the edge weights can be reduced in a similar way as for the Steiner Tree problem. To obtain a PAS, a dynamic program can be used to search for a solution set that is a $(1 + \varepsilon)$-approximation among the precomputed solutions. This step takes $2^{O(k^2)}k^{2^{O(1/\varepsilon)}}n^{O(1)}$ time, and so the overall runtime of the algorithm can be upper bounded by $2^{O(k^2)}n^{2^{O(1/\varepsilon)}}$. This results in the following theorem.

**Theorem 2.6** ([3]). For the Directed Steiner Network problem on planar bidirected graphs a $(1 + \varepsilon)$-approximation can be computed in $2^{O(k^2)}n^{2^{O(1/\varepsilon)}}$ time for any $\varepsilon > 0$, where $k$ is the number of terminals. Moreover, a $(1 + \varepsilon)$-approximate kernel of size $(k/\varepsilon)^{2^{O(1/\varepsilon)}}$ can be computed in polynomial time.

Given that the planar bidirected instances considered for Theorem 2.6 are rather restricted, a natural question becomes (a) whether the runtime can be improved (possibly even to polynomial time, and maybe even an optimum solution can be computed in FPT time), and (b) whether similar algorithms exist for any of the two natural generalizations, i.e., either planar graphs or bidirected graphs. We give partial answers to this question. In particular, the algorithms of Theorem 2.6 are in fact slightly more general than stated: they work even for non-planar bidirected graphs if we want to approximate the optimum planar solution. That is, even if the input graph is bidirected but otherwise unrestricted, the solutions are at most a $(1 + \varepsilon)$-factor more expensive than the cheapest among all planar solutions. Note though that the computed solutions may be non-planar (and could thus even turn out to be cheaper than the optimum planar solution). Considering this more general setting might at first seem rather exotic. However, it turns out that several algorithms found in the literature for Directed Steiner Network on special graph classes have this quality. That is, even if they are stated as algorithms for some input graph class $K$, they can be used to compute solutions in otherwise unrestricted graphs, while the solution quality is compared to the optimum solution from $K$. We give some more examples of this in Section 2.3.

We give negative answers to the above questions in the more general setting just described. This can be interpreted as saying that if these questions can be answered positively in the original setting of planar bidirected graphs, then new algorithmic techniques need to be developed, which are different from those typically found in the literature to date. For the first question on improving the runtime, first off in [3] we prove that it is APX-hard to compute the planar optimum in a bidirected graph. Then, note that the algorithm of Theorem 2.6 is not an EPAS, i.e., the degree of the polynomial factor depends on the approximation factor $\varepsilon$. As we prove in [3], unless FPT=W[1], this dependence is necessary for bidirected inputs where we want to approximate the planar optimum. This also rules out an FPT algorithm to compute the optimum planar solution in bidirected graphs (and hence we used an XP algorithm to obtain Theorem 2.6). For the second question on generalizing the algorithms, we prove in [3] that under Gap-ETH no PAS exists for bidirected input graphs without any further restriction (i.e., when approximating the overall optimum). In [1] we also show that the other obvious generalization, where the input consists of any directed graph and we approximate the planar optimum, has no $(2 - \varepsilon)$-approximation for any $\varepsilon > 0$, under Gap-ETH. As summarized in the following, these results contrast Theorem 2.6.

**Theorem 2.7** ([1; 3]). For the Directed Steiner Network problem the following hardness results hold for any computable function $f$:

1. computing the planar optimum in bidirected input graphs is APX-hard,
2. there is no \( f(k, \varepsilon)n^{O(1)} \) time algorithm that computes a \((1 + \varepsilon)\)-approximation of the planar optimum in bidirected input graphs where \( \varepsilon > 0 \) is part of the input, unless \( \text{FPT} = \text{W}[1] \).

3. there exists a constant \( \alpha > 1 \) such that there is no \( f(k)n^{O(1)} \) time algorithm that computes an \( \alpha \)-approximation to the (overall) optimum in bidirected input graphs, under Gap-ETH.

4. there is no \( f(k)n^{O(1)} \) time algorithm that computes a \((2 - \varepsilon)\)-approximation of the planar optimum in general input graphs, for any \( \varepsilon > 0 \), under Gap-ETH.

2.3 Computing exact solutions in directed graphs

In this section we digress slightly from our main topic of parameterized approximation algorithms, and present some results on computing optimum solutions for special cases of the Directed Steiner Network problem. These results are related to the parameterized approximation algorithms presented in Section 2.2, and are partially also used as subroutines for the latter.

In [10] we analysed the dependence of the parameterized complexity of Directed Steiner Network on the structure of the pattern graphs, as introduced for Theorem 2.5. We proved that the problem is FPT whenever the pattern graphs are restricted to “almost-caterpillars” and \( \text{W}[1] \)-hard otherwise, i.e., we show a dichotomy on the complexity w.r.t. the patterns. Formally these almost-caterpillars are defined as follows, where an out-star or in-star is a directed star for which all edges point away from the center vertex or towards the center vertex, respectively.

**Definition 2.8.** A \( \lambda_0 \)-caterpillar graph is constructed as follows. Take a directed path \((v_1, \ldots, v_{\lambda_0})\) from \( v_1 \) to \( v_{\lambda_0} \), and let \( W_1, \ldots, W_{\lambda_0} \) be pairwise disjoint vertex sets such that \( v_i \in W_i \) for each \( i \in \{1, \ldots, \lambda_0\} \). Now add edges such that either every \( W_i \) forms an out-star with root \( v_i \), or every \( W_i \) forms an in-star with root \( v_i \). A 0-caterpillar is the empty graph. The class \( \mathcal{C}_{\lambda, \delta} \) contains all directed graphs \( H \) such that there is a set of edges \( F \subseteq E(H) \) of size at most \( \delta \) for which the remaining edges \( E(H) \setminus F \) span a \( \lambda_0 \)-caterpillar for some \( \lambda_0 \leq \lambda \). We say that two pattern graphs are translatively equivalent if their transitive closures are isomorphic, and denote by \( \mathcal{C}_{\lambda, \delta}^* \) the class of patterns that are transitively equivalent to some pattern of \( \mathcal{C}_{\lambda, \delta} \).

For example, for the Directed Steiner Tree problem all pattern graphs are in-stars and thus belong to the class \( \mathcal{C}_{1, 0}^* \). For the Strongly Connected Steiner Subgraph problem the patterns can be seen as complete graphs. In this case it turns out that no constants \( \lambda \) and \( \delta \) exist for which the patterns to this problem would belong to some class \( \mathcal{C}_{\lambda, \delta}^* \). Therefore the following theorem in particular recovers the known [DW71; GNS11] complexity results for these two problems. It is much more general though, as it gives a complete dichotomy of the tractability of Directed Steiner Network depending on the structure of the pattern graphs.

**Theorem 2.9 ([10]).** Let \( \mathcal{H} \) be a recursively enumerable class of patterns and let \( k \) be the number of terminals of a given instance.

1. If there are constants \( \lambda \) and \( \delta \) such that \( \mathcal{H} \subseteq \mathcal{C}_{\lambda, \delta}^* \), then Directed Steiner Network restricted to patterns from \( \mathcal{H} \) is FPT for parameter \( k \), and can be solved in \( 2^{O(k + \tau \omega \log \omega)} n^{O(\omega)} \) time, where \( \omega = (1 + \lambda)(\lambda + \delta) \) and \( \tau \) is the vertex cover number of the given input pattern \( H \in \mathcal{H} \).

2. Otherwise, if there are no such constants \( \lambda \) and \( \delta \), then the problem is \( \text{W}[1] \)-hard for parameter \( k \).

To obtain the algorithm of the first part of this theorem, in [10] we show that any optimal solution to a pattern in \( \mathcal{C}_{\lambda, \delta}^* \) has treewidth\(^4\) at most \( 7(1 + \lambda)(\lambda + \delta) \). The algorithm is then

\(^4\)A directed graph has treewidth \( \omega \) if its underlying undirected graph does.
implied by the following useful theorem, which we obtained in [10] via a dynamic programming approach.

**Theorem 2.10 ([10]).** Let an instance of Directed Steiner Network be given by a graph with \( n \) vertices, and a pattern \( H \) on \( k \) terminals with vertex cover number \( \tau \). The cheapest among all solution to \( H \) with treewidth \( \omega \) can be computed in \( 2^{O(k + \tau \omega \log \omega)} n^{O(\omega)} \) time.

This theorem has several consequences, which we now elaborate on. An arbitrary pattern graph \( H \) with \( d \) edges belongs to the class \( C_{0,d}^2 \). Consequently, the optimum solution has treewidth at most \( 7d \) by our results in [10], and the algorithm of Theorem 2.10 can be used to compute the optimum to \( H \) in \( 2^{O(kd \log d)} n^{O(d)} \) time, i.e., we obtain an XP algorithm parameterized by the number of terminals, since \( d < k^2 \). It was actually first shown by Feldman and Ruhl [FR06] that Directed Steiner Network is in XP, and thus our results in [10] recover this fact. Feldman and Ruhl [FR06] however obtain a faster \( n^{O(d)} \) time XP algorithm. Measured in the stronger parameter \( k \), this is an \( n^{O(k^2)} \) time algorithm. As shown by Eiben et al. [Eib+19], this is essentially best possible as no \( f(k)n^{o(k^2/\log k)} \) time algorithm exists for this problem for any computable function \( f \), under ETH. However, as summarized below, in special cases it is possible to beat this lower bound, using Theorem 2.10.

In [3] we show that the treewidth of an optimum planar solution in a bidirected graph is \( O(\sqrt{k}) \), which then implies a faster XP algorithm with runtime \( 2^{O(k^{3/2} \log k)} n^{O(\sqrt{k})} \), as also mentioned in Section 2.2. We also prove [3] that there is no \( f(k)n^{o(\sqrt{k})} \) time algorithm to compute the planar optimum in bidirected graphs, under ETH. For directed planar input graphs, we show in [2] that under ETH no \( f(k)n^{o(k)} \) time algorithm can compute the optimum Directed Steiner Network solution (note that this is a stronger hardness result as the previous one, since here the input graph is planar). Eiben et al. [Eib+19] show that an optimum solution of genus \( g \) has treewidth \( 2^{O(g)} k \) and thus Theorem 2.10 implies an XP algorithm with runtime \( 2^{O(k \log k)} n^{O(k)} \) for solutions of constant genus, matching the previous runtime lower bound. However, for the special case of the Strongly Connected Steiner Subgraph problem, we prove in [2] that the optimum planar solution again has treewidth \( O(\sqrt{k}) \), leading to an XP algorithm with runtime \( 2^{O(k)} n^{O(\sqrt{k})} \). We also obtain [2] a runtime lower bound of \( f(k)n^{o(\sqrt{k})} \) for this problem on planar graphs (which again is a stronger hardness result than previously).

Note that the two algorithms for planar solutions, the one for bounded genus solutions, but also the algorithm of Theorem 2.10, have the quality mentioned in Section 2.2 that they compute optimum planar, bounded-genus, or bounded-treewidth solutions in graphs that have unbounded genus and treewidth.

Another interesting application of Theorem 2.10 is the Strongly Connected Steiner Subgraph problem on bidirected input graphs. While this problem remains NP-hard, in [3] we show that it is FPT parameterized by \( k \), which is in contrast to general input graphs where the problem is \( W[1] \)-hard [GNS11] (as also implied by Theorem 2.9). To show this result, it is not enough to bound the treewidth of a solution and then apply Theorem 2.10 directly, as above for planar optima. In fact, in [3] we give examples in which the optimum solution to Strongly Connected Steiner Subgraph on bidirected graphs has treewidth \( \Theta(k) \). Instead we provide a decomposition of optimum solutions, similar to the theorem of Borchers and Du [BD97] for Steiner Tree or our generalization in Theorem 2.5 for Directed Steiner Network. While the latter two results find sub-graphs that cover a solution (i.e., the sub-graphs may not be edge-disjoint), for Strongly Connected Steiner Subgraph on bidirected graphs we obtain a stronger result, in the sense that a solution can be decomposed into non-overlapping (i.e., edge-disjoint) sub-graphs. Each of these sub-graphs is a solution to some pattern graph \( H \) on the terminals of the input instance, and is a poly-tree, i.e., a directed graph whose underlying
undirected graph is a tree. However, in contrast to Theorem 2.5 the number of terminals in each poly-tree is not bounded by any constant.

The algorithm now proceeds similar to the PAS of Theorem 2.6: it first computes all optimum poly-tree solutions for every possible pattern graph, and then finds the best solution strongly connecting the terminal set by combining the poly-trees using a dynamic program. Since a poly-tree has treewidth 1, Theorem 2.10 can be used to compute an optimum solution to any pattern graph in $2^{O(k)}n^{O(1)}$ time. Furthermore, as the poly-trees of the decomposition are non-overlapping, the algorithm computes an optimum solution to Strongly Connected Steiner Subgraph on bidirected graphs. Note that in contrast to the algorithm of Theorem 2.6 however, computing the solutions to the patterns takes \text{FPT} time, as there is no constant bound on the number of terminals in each poly-tree. This also means that no polynomial-sized kernel is implied by this decomposition.

**Theorem 2.11 ([3]).** The Strongly Connected Steiner Subgraph problem on bidirected graphs is \text{NP}-hard, but can be solved in $2^{k^2+O(k)}n^{O(1)}$ time where $k$ is the number of terminals.

### 3 Clustering

For clustering problems the task is to group the vertices of a metric $(V, \text{dist})$ into sets such that vertices that are close are in the same group, where the closeness is given by some measure depending on the distance function $\text{dist} : V \times V \to \mathbb{R}^+$. Some prominent examples include the $k$-Median, $k$-Center, and Facility Location problems. For each of these, we need to select a subset $F \subseteq V$ of the vertices, called centers or facilities, which act as representatives for the groups. For $k$-Median and $k$-Center the set $F$ can only contain $k$ vertices and we need to minimize $\sum_{v \in V} \text{dist}(v, F)$ and $\max_{v \in V} \text{dist}(v, F)$, respectively, where $\text{dist}(v, F) = \min_{f \in F} \text{dist}(v, f)$. The Facility Location problem essentially is the Lagrangian relaxation of $k$-Median, i.e., there is no bound on the number of facilities but instead each vertex $v \in V$ comes with an opening cost $c(v) \in \mathbb{R}^+$, and we need to minimize $\sum_{f \in F} c(f) + \sum_{v \in V} \text{dist}(v, F)$.

For this thesis we focus on metrics that model transportation networks, given that clustering problems arise in many applications of logistics where, for instance, we would like to place a limited number warehouses or hospitals on a map such that every point is close to one of them. In Section 3.1 we introduce several parameters modelling transportation networks, including the doubling and highway dimensions.

Algorithmically, the $k$-Median and Facility Location problems behave quite differently from the $k$-Center problem. Therefore, we present our results for these problems separately in Sections 3.2 and 3.3, respectively. Our main results in Section 3.2 include a near-linear time approximation scheme for $k$-Median and Facility Location parameterized by the doubling dimension, a PTAS for these problems parameterized by the highway dimension, and also some complementing hardness results for the latter parameter. Additionally, we present some results on metric embeddings of low highway dimension graphs into bounded treewidth graphs. These embeddings imply slower approximation schemes running in quasi-polynomial time for $k$-Median and Facility Location, but in return are applicable to a wider range of problems, including for instance Steiner Tree.

In Section 3.3 we begin with some hardness results for the $k$-Center problem, which show that using either $k$, the doubling dimension, or the highway dimension as a parameter is unlikely to yield better approximation factors than those obtainable in polynomial time. We then consider the combination of the parameter $k$ with either the doubling dimension or the highway dimension, and show that in both cases it is possible to beat the previous lower bounds. Finally, we also show that even when combining all models of transportation networks presented in Section 3.1,
no exact solution for \( k \)-Center can be computed in FPT time, under standard complexity assumptions.

### 3.1 Metrics modelling transportation networks

In this section we present the metrics and parameters used for our results of the following sections, which are mainly based on the structure of transportation networks. For instance, a natural model for road networks is to assume that the given metric is the shortest-path metric of a planar graph, since overpasses and tunnels are relatively rare.

Another reasonable model is to assume that the metric is given by the Euclidean plane, since a road network is embedded on a large sphere (namely the Earth). In cities, where blocks of buildings form a grid of streets, it is reasonable to assume that the distances are given by the Manhattan plane. More generally, one might assume that a transportation network is given by some \( \ell_q \)-norm in \( D \)-dimensional space for some small value of \( D \). That is, the vertices of the given metric \((V, \text{dist})\) are points in \( \mathbb{R}^D \) and the distance function is given by

\[
\text{dist}(u, v) = \left( \sum_{i=1}^{D} |u_i - v_i|^q \right)^{1/q}.
\]

The dimension \( D \) of such a metric space has been studied as a parameter from the parameterized approximation point-of-view *avant la lettre* for quite some time. While it may be unusual to see these results in the light of parameterized algorithms, we specifically do so here in order to obtain a more nuanced view of the complexity of the problems. For instance, it was shown [GGJ77; GI03] that in Euclidean metrics both the k-Median and Steiner Tree problems are paraNP-hard for this parameter (since they are NP-hard even if \( D = 2 \)), and they are APX-hard in general metrics [CC08; JMS02]. However, EPASs for both the Steiner Tree and the k-Median problems in Euclidean metrics were shown to exist in the works of Arora [Aro98] and Koolioopoulos and Rao [KR07], respectively, who showed that a \((1 + \varepsilon)\)-approximation can be computed in \( D^{O(\sqrt{D}/\varepsilon)^{D-1}}n^2 \) time for Steiner Tree and \( 2^{O((\log(1/\varepsilon)/\varepsilon)^{D-1})}D^{O(D)}n^2 \) time for k-Median.\(^5\)

A related setting is the parameterization by the doubling dimension of the underlying metric, which is the smallest integer \( d \) such that any ball \( B_v(r) = \{ u \in V \mid \text{dist}(u, v) \leq r \} \) of radius \( r \) in the metric can be covered by at most \( 2^d \) balls of half the radius \( r/2 \). Any point set in a \( D \)-dimensional \( \ell_q \)-metric has doubling dimension \( O(D) \), and thus the latter parameter generalizes the former. By a result of Talwar [Tal04], there are quasi-polynomial time approximation schemes (QPTASs) for Steiner Tree, k-Median, and Facility Location in metrics of constant doubling dimension \( d \in O(1) \), i.e., they compute a \((1 + \varepsilon)\)-approximation in \( 2^{(\log n)^O(d/e)} \) time for some function \( f \). In the jargon of parameterized algorithms one could classify such an algorithm as a slice-wise quasi-polynomial time approximation scheme. The techniques used to obtain this algorithm for doubling metrics generalize those used for low dimensional \( \ell_q \)-metrics. Since our algorithms presented in Section 3.2 build on these techniques as well, we will introduce them later. Using an entirely different local search technique it is possible to compute a \((1 + \varepsilon)\)-approximation in \( n^{(d/e)O(d)} \) time [FRS19]. This is a PTAS assuming constant doubling dimension \( d \in O(1) \), or in the jargon of parameterized algorithms, it is a slice-wise polynomial time approximation scheme.

A number of our results presented in Sections 3.2 and 3.3 are focussed on the highway dimension, which is a graph parameter specifically formalizing structural properties of transportation networks. We say that a metric has highway dimension \( h \) if it is the shortest-path metric of a graph of highway dimension \( h \). The following definition is taken from [8].

\(^5\)In [Aro98; KR07] the runtimes of these algorithms are stated as \( O(n(\log n)^{O((\sqrt{D}/\varepsilon)^{D-1})}) \) and \( 2^{O((\log(1/\varepsilon)/\varepsilon)^{D-1}}n\log^{D+6}n \), respectively, which can be shown to be upper bounded by \( D^{O((\sqrt{D}/\varepsilon)^{D-1}}n^2 \) and \( 2^{O((\log(1/\varepsilon)/\varepsilon)^{D-1})}D^{O(D)}n^2 \) (see e.g. [KLP19, Lemma 1]).
Definition 3.1. The highway dimension of a graph $G$ is the smallest integer $h$ such that, for some universal constant $c \geq 4$, for every $r \in \mathbb{R}^+$, and every ball $B_v(cr)$ of radius $cr$, there are at most $h$ vertices in $B_v(cr)$ hitting all shortest paths of length more than $r$ that lie in $B_v(cr)$.

The highway dimension was originally defined by Abraham et al. [Abr+10], who specifically restricted the balls to have radius $4r$ in Definition 3.1. They also point out though that the choice of the constant $c$ is somewhat arbitrary. In [8] we prove that when choosing any constant $c$ strictly larger than 4 in Definition 3.1 we obtain additional properties for these graphs, which can be exploited algorithmically (see Section 3.2). Note though that increasing the constant $c$ in Definition 3.1 restricts the class of graphs further. Moreover, as we shown in [8], the highway dimension of a graph according to Definition 3.1 can grow arbitrarily large by just a small change in the constant $c$. Indeed, for any $c$ there is a graph of highway dimension 1 when using $c$ in Definition 3.1, which however has highway dimension $\Omega(n)$ for any constant larger than $c$.

Since the original definition of the highway dimension in [Abr+10], several other definitions (including the above one for larger values of $c$) have been proposed, which we present next. We refer to [Blu19; 8] for detailed discussions.

In a follow-up paper to [Abr+10], Abraham et al. [Abr+16] define a much stronger definition of the highway dimension, which implies that the graphs also have bounded doubling dimension. Hence for this definition, any algorithm that uses the doubling dimension as a parameter can also be used as an algorithm for the highway dimension. Definition 3.1 on the other hand implies metrics of large doubling dimension as noted by Abraham et al. [Abr+10]: a star with unit edge lengths has highway dimension 1 (by using the center vertex to hit all paths), but its doubling dimension is unbounded. While it may be reasonable to assume that road networks have low doubling dimension (which are the main concern in the works of Abraham et al. [Abr+16; Abr+11; Abr+10]), there are metrics modelling transportation networks, for which it can be argued that the doubling dimension is large, while the highway dimension should be small, and thus rather adhere to Definition 3.1: in networks arising from public transportation, longer connections are serviced by larger and sparser stations (such as train stations and airports). More concretely, the so-called hub-and-spoke networks that can typically be seen in air traffic networks is much closer to a star-like network and is unlikely to have bounded doubling dimension, while still having small highway dimension. Thus in these examples it is reasonable to assume that the doubling dimension is a lot larger than the highway dimension.

All definitions of the highway dimension mentioned above imply the existence of sparse shortest path covers, as also introduced by Abraham et al. [Abr+10]. As done in following definition, these can thus be used to define an even more general notion of the highway dimension, which we also use for some of the results presented in Section 3.3. We also show in [8] that this is a strictly more general class, since there are graphs of highway dimension 1 according to Definition 3.2 below, which have unbounded highway dimension according to Definition 3.1 (using the same universal constant $c$). We also note that, as Definition 3.1, the following definition becomes more restrictive the larger the constant $c$ is.

Definition 3.2. Let $c \geq 4$ be a universal constant. For a graph $G$, and $r \in \mathbb{R}^+$, a shortest path cover is a set $\text{SPC}(r) \subseteq V$ of so-called hubs that hit all shortest paths of length in $(r, cr/2]$ of $G$. Such a cover is called locally $s$-sparse for scale $r$, if no ball $B_v(cr/2)$ of radius $cr/2$ contains more than $s$ vertices from $\text{SPC}(r)$. The highway dimension of $G$ is the smallest integer $h$ such that $G$ has a locally $h$-sparse shortest path cover $\text{SPC}(r)$ for every $r \in \mathbb{R}^+$.

Most of our results for low highway dimension metrics presented in the following sections exploit the structure obtained for graphs adhering to the stronger Definition 3.1 using a universal constant $c > 4$. It remains an interesting open question to determine whether there is any
difference in the algorithmic complexity for the studied problems between graphs of highway dimension according to Definition 3.1 and the larger class of graphs adhering to Definition 3.2. Especially on the algorithmic side it is often unclear how to obtain comparable results to those using Definition 3.1 when instead using Definition 3.2.

It should be further noted that (unless otherwise stated above) all the classes of metrics presented in this section are in general incomparable, i.e., they are not contained in one another. In fact, this is even true when comparing to other structural parameters such as shortest-path metrics of graphs with low tree- or pathwidth (cf. [Blu19; 8]).

### 3.2 The $k$-Median problem and its variants

Before turning to our results for the parameters introduced in Section 3.1, we mention some interesting known results for general metrics. Here we also consider the $k$-MEANS problem, which is similar to $k$-MEDIAN but the objective function squares the distances to the centers, i.e., we need to minimize $\sum_{v \in V} (\text{dist}(v, F))^2$. The best approximation ratios achieved by polynomial time algorithms are $2.611 + \varepsilon$ for $k$-MEDIAN [Byr+14], and $9 + \varepsilon$ for $k$-MEANS [Kan+04]. From the hardness side, it is NP-hard to approximate $k$-MEDIAN [JMS02] within a factor $1 + 2/e - \varepsilon \approx 1.73 - \varepsilon$, and $k$-MEANS [Awa+15] within a factor $1 + 8/e - \varepsilon \approx 3.94 - \varepsilon$. While there are some gaps between these results for $k$-MEDIAN and $k$-MEANS, it is an interesting question to ask how the natural parameterization by $k$ changes the approximation ratios for both problems. Cohen-Addad et al. [Coh+19] studied this question and gave exact answers. They show that if we parameterize by $k$, $1 + 2/e$ (for $k$-MEDIAN) and $1 + 8/e$ (for $k$-MEANS) are the exact limits of approximation for parameterized algorithms, giving corresponding upper and lower bounds for this parameter.

In the remainder of this section, we will first present our results for the parametrization by the doubling dimension, where we also give an overview of previous techniques on which ours build. These techniques are then refined for the parametrization by the highway dimension, to which we turn thereafter. Finally, we will also present some alternative techniques to solve problems on low doubling and low highway dimension metrics, which yield slower algorithms but can in return be applied to a wider range of problems.

#### 3.2.1 Low doubling metrics

The starting point of many approximation algorithms for doubling metrics (including Euclidean spaces) is a decomposition of the metric, as presented in the following lemma. Here, a hierarchical decomposition $D$ of a metric $(V, \text{dist})$ is a set of partitions $A_0, A_1, \ldots, A_\lambda$ of $V$, where $A_i$ refines $A_{i+1}$, i.e., every part $A \in A_i$ is contained in some part of $A_{i+1}$. Moreover, in $A_0$ every part contains a singleton vertex, while $A_\lambda$ contains only one part, namely $V$. For a point $v \in V$ and a radius $r > 0$, we say that the ball $B_v(r)$ is cut by $D$ at level $i$ if $i$ is the largest integer for which the ball $B_v(r)$ is not contained in a single part of $A_i$. The aspect ratio of a metric $(V, \text{dist})$ is the largest distance divided by the shortest distance of any points in $V$.

**Lemma 3.3** (Reformulation of [BG13; Tal04]). For any metric $(V, \text{dist})$ of doubling dimension $d$ and aspect ratio $\alpha$, and for any $\rho > 0$, there exists a polynomial-time computable randomized hierarchical decomposition $D = \{A_0, \ldots, A_{\log_2 \alpha}\}$ such that:

1. **Scaling probability:** for any $v \in V$, radius $r$, and level $i$, we have
   \[ \Pr[D \text{ cuts } B_v(r) \text{ at level } i] \leq 2^{O(d)} \cdot r/2^i. \]

2. **Portal set:** every part $A \in A_i$ where $A_i \in D$ comes with a set of portals $P_A \subseteq A$ that is
(a) **concise**: the size of the portal set is bounded by $|P_A| \leq 1/\rho^d$, and 
(b) **precise**: for every node $u \in A$ there is a portal $p \in P_A$ with $\text{dist}(u, p) \leq \rho^{2i+1}$.

We briefly sketch the standard use of this decomposition (see [Aro98; Mit99; Tal04]). For clustering problems, one can show that there exists a *portal-respecting solution* with near-optimal cost. In this structured solution, each client connects to a facility via a *portal-respecting path* that enters and leaves any part $A$ of $D$ only through a node of the portal set $P_A$. These portals therefore act as separators of the metric. A standard dynamic program approach can then compute the best portal respecting solution.

To ensure that there is a portal-respecting solution with near-optimal cost, one uses the preciseness property of the portal set: the additional distance (referred to as *distortion*) of connecting a client $c$ with a facility $f$ through portals instead of directly is bounded as follows. Let $i$ be the level at which $D$ cuts $c$ and $f$, meaning that $i$ is the maximum integer for which $c$ and $f$ lie in different parts of $A_i$. At every level $j \leq i$, the distortion incurred by making a detour to the closest portal is $O(\rho 2^j)$, due to the triangle inequality and the preciseness of the portal set. Hence the total distortion is $\sum_{j \leq i} O(\rho 2^j) = O(\rho 2^i)$. Now, the bound on the scaling probability of the decomposition ensures that $c$ and $f$ are cut at level $i$ with probability $2^{O(d)} \cdot \text{dist}(c, f)/\rho^i$. Hence combining these two bounds over all $[\log_2 \alpha] + 1$ levels ensures that, in expectation, the distortion between $c$ and $f$ is bounded by $2^{O(d)} \cdot \text{dist}(c, f) /\rho \cdot \log_2 \alpha$. Using standard preprocessing techniques one can ensure that the aspect ratio is $\alpha = O(n/\varepsilon)$ when aiming for a $(1 + \varepsilon)$-approximation. Hence choosing $\rho = \frac{\varepsilon}{2^{O(d)} \log(n/\varepsilon)}$ gives a distortion of $\varepsilon \cdot \text{dist}(c, f)$. Summing over all clients proves that there exists a near-optimal portal-respecting solution.

The issue with this approach is that to obtain this level of preciseness, according to the conciseness property the number of needed portals for each part $A$ of the decomposition $D$ is $\left(\frac{\log(n/\varepsilon)}{\varepsilon}\right)^{O(d)}$, and the dynamic program has a runtime that is exponential in this number. As Talwar [Tal04] showed, the resulting algorithm runs in $2^{\left(d \log \frac{n}{\varepsilon}\right)^{O(d)}}$ time, i.e., for any constant $d \in O(1)$ we obtain a QPTAS. However, in some cases one can lower the number of portals per part needed and thus obtain a PTAS. In Euclidean space for example, the celebrated “patching lemma” [Aro98] shows that only a constant number (depending on $\varepsilon$) of portals are needed for the Steiner Tree. Similarly, Kolliopoulos and Rao [KR07] showed that for $k$-Median in Euclidean space only a constant number of portal are needed, if one uses a slightly different decomposition of the metric. Surprisingly, obtaining such a result for doubling metrics is much more challenging.

A second challenge occurs when trying to solve problems such as $k$-Means, where the objective function squares the distances to the centers. In this case, the analysis of Arora [Aro98], Mitchell [Mit99], and Talwar [Tal04] does not apply: if two points are separated at a high level of the decomposition, then making a detour to the closest portal may incur an expected cost much higher than the cost of the optimal solution.

In [4] we show how to circumvent these issues for clustering problems in low doubling metrics. Our contribution can be viewed as a “patching lemma” for problems such as $k$-Median, $k$-Means, and Facility Location. Namely, we present an approach which (1) reduces the number of portals to a constant, (2) works for any clustering objective which is defined as the sum of distances to some constant $q$ (with $k$-Median, Facility Location, and $k$-Means as prominent special cases), and (3) works not only for Euclidean but also for doubling metrics.

To achieve this, in [4] we show how to reduce the number of levels on which a client can be cut from its facility. For this, we present a processing step of the instance that helps deal with clients cut from their facility at a high level. Roughly speaking, our algorithm first computes a constant factor approximation $L$, and a client $c$ is called *badly-cut* if the decomposition $D$ cuts it from its closest facility of $L$ at a level larger than $\log(\text{dist}(c, L)/\varepsilon)$. Every badly-cut client is moved to its
closest facility of \( L \). Moreover, every client at distance less than \( \varepsilon \cdot \text{dist}(c, L) \) of its closest facility of \( L \) can be moved to it as well. It is then shown that this new instance \( \mathcal{I}_D \) has small distortion, which essentially means that any solution to \( \mathcal{I}_D \) can be converted to a solution of the original instance \( \mathcal{I} \) while only losing a \((1 + \varepsilon)\)-factor in quality. In this instance \( \mathcal{I}_D \), all clients are cut from their closest facility of \( L \) at some level between \( \log(\varepsilon \cdot \text{dist}(c, L)) \) and \( \log(\text{dist}(c, L)/\varepsilon) \). Using this property, we show that \( c \) and its closest center in the optimal solution are also cut at a level in that range. As there are only \( O(\log(1/\varepsilon)) \) levels in this range, by the previous arguments it now suffices to set \( \rho = 2\varepsilon^{2/d \cdot \log(1/\varepsilon)} \) for the hierarchical decomposition in order to get a near-optimal portal-respecting solution. As this implies a number of portals that only depends on the doubling dimension \( d \) and the approximation factor \( \varepsilon \), according to the conciseness property we obtain the following randomized approximation schemes parameterized by \( d \) and \( \varepsilon \). In particular, we get a significant improvement from the previously fastest known slice-wise polynomial time approximation schemes as given by [FRS19], to randomized EPASs with near-linear running time.

**Theorem 3.4** ([4]). For the \( k \)-Median, \( k \)-Means, and Facility Location problems on metrics of doubling dimension \( d \) a \((1 + \varepsilon)\)-approximation can be computed in \( \tilde{O}(2^{(1/\varepsilon)}O(d^2) n) \) time with success probability \( 1 - \tilde{O}(\varepsilon) \) for any \( \varepsilon > 0 \).

It is interesting to note that the double-exponential dependence on \( d \) in the runtime cannot be improved to single-exponential, since any metric has doubling dimension \( O(\log n) \) but the problems are APX-hard [Awa+15; GK99; JMS02] in general metrics. As mentioned before, Theorem 3.4 also holds for the corresponding problems where the distances in the objective functions are raised to the power of any integer \( q \), so that \( k \)-Median and \( k \)-Means for instance are the special cases where \( q = 1 \) and \( q = 2 \), respectively. Furthermore, the techniques can be generalized to obtain bicriteria approximation schemes with similar running times for prize-collecting and outlier versions of the problems (cf. [4]).

### 3.2.2 Low highway dimension graphs

We now turn to the parametrization by the highway dimension, for which in [12] we obtain slower approximation schemes for clustering problems than those given by Theorem 3.4 for the doubling dimension. Nevertheless, our algorithms for the highway dimension utilize the techniques described above for the doubling dimension, in addition to some structural insights we obtained for low highway dimension graphs in [8].

More concretely, the above arguments for doubling metrics hold thanks to the hierarchical decomposition given by Lemma 3.3. It is therefore tempting to try to devise a similar decomposition for metrics of low highway dimension. However, it turns out that while these metrics have some similarities to low doubling metrics, they behave very differently, so that we were not able to obtain a decomposition with the properties given by Lemma 3.3. Instead, in [8] we introduce the following town decomposition of low highway dimension metrics, which gives a formal connection to doubling metrics. We obtain its properties specifically for Definition 3.1 of the highway dimension using any universal constant \( c \) strictly larger than 4. In the following, a child part of a part \( A \in \mathcal{A}_i \) of some hierarchical decomposition \( \mathcal{D} = \{ \mathcal{A}_0, \ldots, \mathcal{A}_\lambda \} \) is a part \( A' \in \mathcal{A}_i \) on the level below \( i \) for which \( A' \subseteq A \).

**Theorem 3.5** ([8]). Given \( \rho > 0 \) and a shortest-path metric \((V, \text{dist})\) of highway dimension \( h \) according to Definition 3.1 for any universal constant \( c > 4 \), there exists a polynomial-time computable deterministic hierarchical decomposition \( \mathcal{T} \), called the town decomposition, such that every part \( T \in \mathcal{T} \), called a town, has a set of hubs\(^6\) \( X_T \subseteq T \) with the following properties:

\(^6\)called approximate core hubs in [8].
The hub set $X_T$ is similar to the portal set of Lemma 3.3, but has some fundamental differences. First note that the preciseness property of Theorem 3.5 is different from the preciseness of Lemma 3.3: in the former, some hub $x$ is guaranteed to be close to a shortest path between $u$ and $v$ (and could thus be far from each of $u$ and $v$), while in Lemma 3.3 the portals lie close to all vertices. Secondly, the town decomposition is deterministic, and so it may happen that a client and its facility are cut at a very high level relative to their distance — something that happens only with small probability in the doubling setting thanks to the scaling probability. Another main difference is that the size of $X_T$ might be unbounded. As a consequence, it cannot be directly used as a portal set in a dynamic program with sub-exponential runtime. To deal with this, in [12] we combine the town decomposition with a hierarchical decomposition of each set $X_T$ according to Lemma 3.3, to build a decomposition of low highway dimension graphs more akin to Lemma 3.3, as stated in the following lemma.

**Lemma 3.6** ([12]). Given $\rho > 0$ and a metric $(V, \text{dist})$ with aspect ratio $\alpha$ where $(V, \text{dist})$ is a shortest-path metric of a graph with highway dimension $h$ according to Definition 3.1 for any universal constant $c > 4$, there exists a polynomial-time computable randomized hierarchical decomposition $D = \{A_0, \ldots, A_{\log_2 \alpha}\}$ of $V$ such that:

1. **Scaling probability:** for any $v \in V$, radius $r$, and level $i$, we have
   \[
   \Pr[D \text{ cuts } B_v(r) \text{ at level } i] \leq (h \log(1/\rho))^{O(1)} \cdot r/2^i.
   \]

2. **Interface:** for any $A \in A_i$ on level $i \geq 1$ there exists an interface $I_A \subseteq V$, which is
   (a) **concise:** $|I_A| \leq (h/\rho)^{O(1)}$, and
   (b) **precise:** for any $u, v \in A$ such that $u$ and $v$ are cut by $D$ at level $i - 1$, there exists $p \in I_A$ with $\text{dist}(u, p) + \text{dist}(p, v) \leq \text{dist}(u, v) + 34 \cdot \rho 2^i$.

As a consequence of using the town decomposition of Theorem 3.5 to construct the hierarchical decomposition of Lemma 3.6, a notable difference to the portals of Lemma 3.3 is that the preciseness property of the interface in Lemma 3.6 is weaker: as for Theorem 3.5, the hubs can be far from some vertices as long as they lie close to the shortest path. As a consequence, no analogue of near-optimal portal-respecting paths exist as was the case for portals. Instead, when connecting a client $c$ with a facility $f$ we need to use the interface point $p \in I_A$ of a part $A$ containing both $c$ and $f$, such that $p$ lies close to the shortest path between $c$ and $f$. This shifts the perspective from externally connecting vertices of a part to vertices outside a part, as done for portals, to internally connecting vertices of parts, as needs to be done for interfaces.

As a consequence, in [12] we develop a dynamic program, which follows more or less standard techniques as for instance given in [ARR98; KR07], but needs to handle the weaker preciseness property of the interface. The main idea is to guess the distances from interface points to facilities while recursing on the decomposition $D$ of Lemma 3.6. The runtime of this algorithm is thus exponential in the number of interface points. Thanks to our techniques developed in [4] as described above, we can assume that this number is constant for clustering problems. However, due to the shifted perspective towards internally connecting vertices of parts, the runtime of the dynamic program also is exponential in the total number of levels. As for doubling metrics the aspect ratio can be reduced [8] to $\alpha = O(n/\varepsilon)$ when aiming for a $(1 + \varepsilon)$-approximation, and so the number of levels of the decomposition is logarithmic in the input size. This implies that the runtime is polynomial, and we obtain the following theorem.
**Theorem 3.7** ([12]). *For the k-Median, k-Means, and Facility Location problems on metrics of highway dimension \( h \) according to Definition 3.1 for any universal constant \( c > 4 \), a \((1 + \varepsilon)\)-approximation can be computed in \( n^{(h/\varepsilon)^{O(1)}} \) time with success probability \( 1 - O(\varepsilon) \) for any \( \varepsilon > 0 \).*

Since the techniques for Theorem 3.4 are also used for Theorem 3.7, similar to the former the algorithm of Theorem 3.7 generalizes to the corresponding problems where the distances in the objective function are raised to the power of any integer \( q \), and bicriteria approximation schemes with similar runtimes can be obtained for prize-collecting and outlier versions. Note though that in contrast to Theorem 3.4 the algorithm of Theorem 3.7 is not an EPAS, but rather a slice-wise polynomial time approximation scheme. Whether an EPAS or even a PAS exists for the parameterization by the highway dimension as well, remains an intriguing open question.

In [12] we show that, unless \( \text{P}=\text{NP} \), no polynomial time algorithm exists to compute optimum solutions, even in the most restrictive case when the highway dimension is 1. In fact, we obtained similar results in [5] for the Steiner Tree and Travelling Salesman problems, where for the latter we are given a metric and we need to find the shortest tour that visits all vertices. These hardness results are valid regardless of which definition of the highway dimension is used (in particular Definition 3.1 can be made arbitrarily strong by using any universal constant \( c \geq 4 \)).

**Theorem 3.8** ([5; 12]). *The k-Median, k-Means, Facility Location, Steiner Tree, and Travelling Salesman problems are NP-hard on metrics of highway dimension 1 according to Definition 3.1 for any universal constant \( c \geq 4 \).*

### 3.2.3 Metric embeddings

We now slightly digress from the topic of this section by considering not just clustering problems but other problems as well. We present an alternative view on hierarchical decompositions leading to so-called embeddings. This tool is rather general and can be used to obtain approximation schemes for clustering problems, but also others such as Steiner Tree and Travelling Salesman, albeit with in larger running times compared to the algorithms presented so far. The idea is to map a given metric into another metric on the same vertex set, which on one hand slightly distorts the distances, but on the other hand introduces some structural properties that can be exploited algorithmically. If the problem at hand can then be solved on the latter metric, then the approximation factor is determined by the distortion when mapping the solution back to the input metric. We will specifically be focussing on mappings into shortest-path metrics of graphs with small treewidth, since plenty of algorithms are known for such graphs. Such a mapping can be probabilistic and is defined as follows.

**Definition 3.9.** Let \((V, \text{dist})\) be a metric and let \( \mathcal{E} \) be a distribution over metrics \((V, \text{dist}')\) on the same vertex set \( V \). If for all \( u, v \in V \), \( \text{dist}(u, v) \leq \text{dist}'(u, v) \) for each \( \text{dist}' \in \mathcal{E} \), and \( E_{\text{dist}' \in \mathcal{E}} [\text{dist}'(u, v)] \leq a \cdot \text{dist}(u, v) \), then \( \mathcal{E} \) is an *embedding* with (expected) stretch or distortion \( a \). If every \( \text{dist}' \in \mathcal{E} \) is the shortest-path metric of some graph class \( \mathcal{G} \), then \( \mathcal{E} \) is a (probabilistic) embedding into \( \mathcal{G} \).

It was noted by Talwar [Tal04] that Lemma 3.3 implies a polynomial-time computable probabilistic embedding of any metric of doubling dimension \( d \) and aspect ratio \( \alpha \) into graphs of treewidth \( (d \log(\alpha)/\varepsilon)^{O(d)} \) with expected distortion \( 1 + \varepsilon \). More concretely, to compute a graph of low treewidth from a given metric, first the hierarchical decomposition \( D = \{ A_0, \ldots, A_{\lceil \log_2 \alpha \rceil} \} \) of Lemma 3.3 is computed. The graph then contains all edges between portals \( P_A \) for each part \( A \), and also all edges connecting the portals \( P_A \) with the portals \( P_{A'} \) of each child part \( A' \) of \( A \). The weight of such an edge \( uv \) is simply the distance between \( u \) and \( v \) given by the metric.
In particular, any portal-respecting path of the metric exists in the graph, and so the distortion is $1 + \varepsilon$ if $\rho$ is set to an appropriate value for Lemma 3.3, as previously argued in Section 3.2.1. At the same time, a tree decomposition of the graph can be obtained by using the tree structure of $D$ (i.e., every part corresponds to a node connected to the nodes of its child parts) and introducing a bag for each part $A$ that contains its portal set $P_A$ and the portal sets $P_{A'}$ of all its child parts $A'$. For doubling metrics it can be ensured [4; Tal04] that the portal sets are nested, which means that every portal of $A$ which happens to be in one of its child parts $A'$ is also a portal of $A'$, i.e., $P_A \cap A' \subseteq P_{A'}$. Due to this, we can prove that we obtain a valid tree decomposition. Furthermore, it can be shown that each part has at most $2^{O(d)}$ child parts, and thus the treewidth is bounded by the conciseness property of Lemma 3.3, which bounds the size of the portal sets.

In [8] we build on this construction to show that also low highway dimension metrics can be embedded into graphs of bounded treewidth. A key ingredient for this is again our structural insight into low highway dimension graphs given by Theorem 3.5. On a high level, a graph of bounded treewidth is constructed by computing an embedding of each hub set $X_T$ for every town $T$ of a town decomposition of the given metric. Since Theorem 3.5 guarantees that each set $X_T$ has bounded doubling dimension, each individual embedding has bounded treewidth, as argued above. The challenge now is to combine all of these embeddings into one, while making sure that both the distortion and the treewidth are still small. One problem for instance is that hub sets $X_T$ of different towns are not nested, i.e., it may happen that a hub of a town $T$ is not a hub of the child town, but then is again a hub of some lower-level descendent town of $T$. Furthermore, there is no bound on the number of child towns of a given town. For this and other reasons that distinguish low highway from low doubling dimension metrics, compared to low doubling metrics a lot more work goes into proving the following theorem.

**Theorem 3.10 ([8]).** Let $(V, \text{dist})$ be a metric with aspect ratio $\alpha$ where $(V, \text{dist})$ is a shortest-path metric of a graph with highway dimension $h$ according to Definition 3.1 for any universal constant $c > 4$. For any $\varepsilon > 0$, there is a polynomial-time computable probabilistic embedding of $(V, \text{dist})$ with expected distortion $1 + \varepsilon$ into graphs of treewidth $(\log \alpha)^{O(\log^2 (h/\varepsilon))}$.

Using known algorithms for bounded treewidth graphs on the embedding given by Theorem 3.10, we obtain randomized approximation schemes for metrics of low highway dimension for problems such as $k$-MEDIAN and FACILITY LOCATION, but also STEINER TREE and TRAVELLING SALESMAN. The expected approximation guarantee is given by the distortion of the distances. To bound the runtime, as previously we may preprocess the metric (cf. [8]) so that its aspect ratio is $O(n/\varepsilon)$, which means that the treewidth bound of Theorem 3.10 is poly-logarithmic for metrics of constant highway dimension $h \in O(1)$ and for constant approximation factors $\varepsilon \in \Theta(1)$. Since algorithms for bounded treewidth graphs have running times exponential in the treewidth, this gives slice-wise quasi-polynomial runtimes, i.e., we obtain QPTASs for the given problems via the embedding of Theorem 3.10. It remains open whether QPTASs also exist for these problems when using the more general Definition 3.2 for the highway dimension.

Since the embedding for low doubling metrics by Talwar [Tal04] was constructed using the hierarchical decomposition of Lemma 3.3, a natural question is whether the above embedding for low highway dimension metrics can be simplified by using the corresponding decomposition of Lemma 3.6. This seems plausible, and might even yield improved bounds on the treewidth compared to Theorem 3.10. However this still needs to be explored and, as of writing this thesis, is left for future work. Similar to obtaining the PTAS of Theorem 3.7 based on Lemma 3.6, one main challenge for such an embedding using Lemma 3.6 is the non-existence of portal-respecting paths, which are used in the construction for Theorem 3.10 to bound the distortion.

The algorithms that are used on the metric embedding resulting from Theorem 3.10 to obtain...
QPTASs for low highway dimension graphs can be any FPT or XP algorithms parameterized by the treewidth \( t \), of which the literature provides plenty. Specifically for Steiner Tree and Travelling Salesman, there are rather efficient single-exponential \( 2^{O(t)}n^{O(1)} \) time FPT algorithms for this parameter [Bod+15]. We show in [5] that this implies the curious fact that these problems are weakly NP-hard on graphs of the smallest possible highway dimension, i.e., on graphs of highway dimension 1 the problems are NP-hard due to Theorem 3.8, but they also admit fully polynomial time approximation schemes (FPTASs), which compute a \((1 + \varepsilon)\)-approximation with a runtime that is polynomial in both the input size and \( \varepsilon \).

We prove this in [5] as follows. First we show that graphs of highway dimension 1 have treewidth bounded in their aspect ratio \( \alpha \) (which can be seen as a trivial embedding with distortion 1), even when using the more general Definition 3.2. As before, we may reduce the aspect ratio to \( O(n/\varepsilon) \), and thereby lose a \((1 + \varepsilon)\)-factor in the solution quality for Steiner Tree and Travelling Salesman. Thus according to the following theorem, the single-exponential time FPT algorithms for parameter treewidth [Bod+15] run in \( 2^{O(\log(n/\varepsilon))}n^{O(1)} = (n/\varepsilon)^{O(1)} \) time on the reduced instances.

**Theorem 3.11 ([5]).** Let \( G \) be a graph with aspect ratio \( \alpha \) and highway dimension 1 according to Definition 3.2 for any universal constant \( c \geq 4 \). The treewidth of \( G \) is \( O(\log \alpha) \).

In general, one might hope to prove similar bounds on the treewidth of graphs with of highway dimension larger than 1, i.e., one might conjecture that any graph of highway dimension \( h \) has treewidth, say, \((h \log \alpha)^{O(1)}\) or \( O(\log^2 \alpha) \). Such a bound would make it possible to circumvent the rather involved construction of the embedding given by Theorem 3.10. Also, depending on the quality of the bound this might imply faster approximation schemes for Steiner Tree and Travelling Salesman due to the single-exponential FPT algorithms for parameter treewidth. However, we can exclude at least some such general treewidth bounds for graphs of low highway dimension using a result we obtained in [7], which consists of an embedding of low doubling metrics into graphs of bounded highway dimension. Since this embedding is used to obtain lower bounds, it is interesting to note that it can be applied with the more restrictive Definition 3.1 of the highway dimension, and moreover the embedding is deterministic.

**Theorem 3.12 ([7]).** Let \((V, \text{dist})\) be a metric with aspect ratio \( \alpha \) and doubling dimension \( d \). For any \( \varepsilon > 0 \), there is a polynomial-time computable deterministic embedding of \((V, \text{dist})\) with distortion \( 1 + \varepsilon \) into a graph of highway dimension \( O((\log(\alpha)/\varepsilon)^d) \) according to Definition 3.1 for any universal constant \( c \geq 4 \).

This embedding implies a lower bound excluding a treewidth of \((h \log \alpha)^{O(1)}\) for graphs of highway dimension \( h \) and aspect ratio \( \alpha \), as follows. We start from a metric given by a regular \( k \times k \) grid in the plane endowed with the \( \ell_1 \)-norm, which has doubling dimension 2 and aspect ratio \( k \). Using Theorem 3.12, from this we obtain a graph \( G \) of highway dimension \( O((\log(k)/\varepsilon)^2) \). Now, consider two neighbouring nodes (at distance 1) in the input grid, and note that connecting them using any additional nodes gives a path of length at least 3 due to the \( \ell_1 \)-norm. Thus if two neighbouring nodes end up not being connected by an edge in \( G \), then any path between these vertices in \( G \) has length at least 3. Setting \( \varepsilon < 2 \) to obtain a distortion of less than 3 ensures that \( G \) contains a \( k \times k \) grid as a subgraph, since the embedding of Theorem 3.12 is deterministic so that every pair of neighbouring nodes of the grid must end up being connected by an edge in \( G \). Hence for instance setting \( \varepsilon = 1 \) we obtain a graph with treewidth \( \Omega(k) \), aspect ratio \( O(k) \), and highway dimension \( O(\log^2 k) \). This excludes a treewidth bound of the form \((h \log \alpha)^{O(1)}\).

Note that a treewidth of \( O(\log^2 \alpha) \) is not excluded by the above argument. However it still seems unlikely due to the following. In [7] we use Theorem 3.12 to prove that, unless P=NP, the \( k \)-Center problem has no polynomial time \( (2 - \varepsilon) \)-approximation algorithm on graphs
with highway dimension $O(\log^2 n)$ (cf. Section 3.3). However, we conjecture that the same inapproximability should hold for graphs of highway dimension $O(1)$. If this is true, then we could again use standard preprocessing to reduce the aspect ratio to $O(n/\varepsilon)$, from which we would obtain a treewidth bound of $O(\log^h \alpha) = O(\text{polylog}(n/\varepsilon))$ for such graphs while distorting the distances by a factor of $1 + \varepsilon$. Now, for $k$-CENTER Katsikarelis et al. [KLP19] obtain an EPAS with runtime $t^{O(1/\varepsilon)} n^{O(1)}$ parameterized by the treewidth $t$, which would run in quasi-polynomial time on graphs with treewidth $O(\text{polylog}(n/\varepsilon))$. Consequently, a treewidth bound of $O(\log^h \alpha)$ for graphs of highway dimension $h$ in combination with a reduction to show APX-hardness for graphs of highway dimension $O(1)$, would imply quasi-polynomial time algorithms to solve NP-hard problems. However under standard complexity assumptions this is not possible.

3.3 The $k$-Center problem

The $k$-CENTER problem is harder to approximate than $k$-MEDIAN and its variants. In particular, by a result of Hochbaum and Shmoys [HS86], the $k$-CENTER problem on general input graphs has a polynomial time 2-approximation algorithm, but this approximation factor is also best possible, unless P=NP. The lower bound of 2 on the approximation factor is rather notorious as it remains valid for many special cases and parameterizations of the problem, including those introduced in Section 3.1 as models for transportation networks. We summarize this in the following theorem.

Theorem 3.13. The $k$-Center problem has no $(2 - \varepsilon)$-approximation algorithms for any $\varepsilon > 0$ in the following cases and runtimes:

- on general graphs in $f(k)n^{O(1)}$ time [7] (i.e., parameterized by the number of centers $k$) for any computable function $f$, unless W[2]=FPT,
- on planar graphs in polynomial time [Ple80], unless P=NP,
- on two-dimensional Manhattan metrics (which have doubling dimension 2) in polynomial time [FG88], unless P=NP,
- on graphs of highway dimension $O(\log^2 n)$ according to Definition 3.1 for any universal constant $c \geq 4$ in polynomial time [7], unless P=NP,
- on graphs of highway dimension $h$ according to Definition 3.1 for any universal constant $c \geq 4$ in $2^{2^{o(\sqrt{h})}} \cdot n^{O(1)}$ time [7], under ETH.

The last two lower bounds of Theorem 3.13 for the highway dimension are based on the embedding of Theorem 3.12, which we use in [7] to reduce the problem on metrics of doubling dimension 2 to graphs of highway dimension $O(\log^2 n)$. The hardness of approximation for the former (as also stated in Theorem 3.13) thus carries over to the latter. Note that the hardness for graph of highway dimension $O(\log^2 n)$ does not rule out $(2 - \varepsilon)$-approximation algorithms parameterized by the highway dimension, and this is left as an open problem. However, under ETH, the same reduction implies that if such an algorithm exists, then its running time must be enormous, as it must be at least doubly exponential, as stated in Theorem 3.13.

In conclusion of Theorem 3.13 it seems that considering any model of Section 3.1 for transportation networks or even the parametrization by the number of centers $k$, does not help to overcome the polynomial-time $(2 - \varepsilon)$-inapproximability that the $k$-CENTER problem exhibits for general inputs. However, combining $k$ as a parameter with any of the models it is possible to beat this lower bound. For instance, by a result of Fox-Epstein et al. [FKS19] the $k$-CENTER problem on edge-weighted planar graphs admits an efficient polynomial-time bicriteria approximation scheme, which for any $\varepsilon > 0$ and some function $f$ computes a solution in $f(\varepsilon)n^{O(1)}$ time that uses at most $(1 + \varepsilon)k$ centers and approximates the optimum with at most $k$ centers within a factor of $1 + \varepsilon$. This algorithm implies an EPAS for parameter $k$ on planar graphs, since
for instance setting \( \varepsilon = \min\{\varepsilon', \frac{1}{2k}\} \) forces the algorithm to compute a \((1 + \varepsilon')\)-approximation in \( f(k, \varepsilon')n^{O(1)} \) time using at most \((1 + \varepsilon)k \leq k + \frac{1}{2} \) centers, i.e., at most \( k \) centers as \( k \) is an integer.

Another example is given by Agarwal and Procopiuc [AP02] who showed that for any \( \ell_q \) metric in \( D \) dimensions, the \( k\text{-Center} \) problem has an EPAS when combining \( k \) and \( D \) as parameters. In [11] we generalize the latter to any metric of doubling dimension \( d \). Our algorithm first guesses the optimum cost \( C \). It then greedily computes a so-called \( \delta\text{-net} \), which is a subset of the vertices of the given metric such that every vertex is at distance at most \( \delta \) from some net point, while all net points are at distance more than \( \delta \) from each other. Any \( \delta\text{-net} \) of a metric with doubling dimension \( d \) can be shown to have size at most \( k/\varepsilon O(d) \) when setting \( \delta = \Theta(\varepsilon C) \). Thus we may compute an optimum \( k\text{-Center} \) solution of the vertices of the \( \delta\text{-net} \) by brute-force in FPT time. The distance properties of the \( \delta\text{-net} \) for \( \delta = \Theta(\varepsilon C) \) imply that this solution is a \((1 + \varepsilon)\)-approximation in the input metric, and thus we obtain the following theorem.

**Theorem 3.14** ([11]). Given \( \varepsilon > 0 \) and a metric of doubling dimension \( d \), a \((1 + \varepsilon)\)-approximation for \( k\text{-Center} \) can be computed in \((k^{k}/\varepsilon O(kd))n^{O(1)} \) time.

For low highway dimension graphs, Becker et al. [BKS18] used the structural results found in Theorem 3.5 to show that when using Definition 3.1 for any universal constant \( c > 4 \) there is an EPAS for \( k\text{-Center} \) parameterized by \( k \) and the highway dimension \( h \). For the more general Definition 3.2 of the highway dimension it is not known whether a PAS exists for this parameterization. However, in [7] we present a 3/2-approximation for \( k\text{-Center} \), which combines \( k \) and this notion of highway dimension as parameters. Hence also for the more general class of graphs given by Definition 3.2 the notorious \((2 - \varepsilon)\)-inapproximability of the \( k\text{-Center} \) problem can be beaten.

**Theorem 3.15** ([7]). Given \( \varepsilon > 0 \) and a metric of highway dimension dimension \( h \) according to Definition 3.2 for any universal constant \( c \geq 4 \), a 3/2-approximation for \( k\text{-Center} \) can be computed in \( 2^{O(kh \log h)}n^{O(1)} \) time.

As we show in [8] some fundamental properties needed to prove the structure given by Theorem 3.5 break down when using the more general Definition 3.2 for the highway dimension. Thus, in contrast to the EPAS of Becker et al. [BKS18], we cannot rely on Theorem 3.5 to prove Theorem 3.15. Using the weaker Definition 3.2 for the highway dimension still implies some interesting structure though: fixing any value \( r \) and a shortest path cover \( \text{SPC}(r) \), we show in [7] that the vertices of the metric are either at distance at most \( r \) from some hub of \( \text{SPC}(r) \), or they lie in clusters\(^7\) of diameter at most \( r \) that are at distance more than \( 2r \) from each other.

\(^7\)in fact these clusters are similar to (but not quite the same as) the towns in the town decomposition of Theorem 3.5 (cf. [8]).
(see Figure 1). Hence, given the cost $C$ of the optimum $k$-Center solution, for $r = C/2$ a center that resides in a cluster cannot cover any vertices of some other cluster. In this sense the clusters are “independent” of each other. At the same time we are able to bound the number of hubs in $\text{spc}(C/2)$ in terms of $k$ and the highway dimension. Roughly, this is comparable to graphs with small vertex cover, since the vertices that are not part of a vertex cover form an independent set. In this sense the highway dimension is a generalization of the vertex cover number, and this is in fact the reason why computing the highway dimension is NP-hard, as we show in [8].

At the same time the $k$-Center problem is a generalization of the Dominating Set problem, where we need to select the smallest number of vertices in a graph such that every vertex is at hop-distance at most 1 from a selected vertex. This problem is $\text{W}[2]$-hard [DF13], but it is FPT using the vertex cover number as the parameter [Alb+02]. This is one of the reasons why combining the two parameters $k$ and $h$ yields a parameterized 3/2-approximation algorithm for $k$-Center. In fact the similarity seems so striking at first that one is tempted to reduce the problem of finding a 3/2-approximation for $k$-Center on low highway dimension graphs to solving Dominating Set on a graph of low vertex cover number. However, it is unclear how this can be made to work. Instead, in [7] we devise an involved algorithm that is driven by the intuition that the two problems are similar to obtain Theorem 3.15.

Given the above parameterized approximation algorithms for models of transportation networks when combining with the parameter $k$, a natural question is whether it is actually necessary to approximate in these cases. That is, can we hope to compute the optimum $k$-Center solution in comparable running times? In [11] we study this question and conclude that under standard complexity assumptions this is not possible, even if we combine all the models found in Section 3.1, as formalized by the following theorem. It is interesting to note that all the mentioned graph classes and parameters are incomparable to each other, as we show in [8] and discussed in more detail by Blum [Blu19].

**Theorem 3.16 ([11]).** Even on edge-weighted planar graphs of doubling dimension $O(1)$, the $k$-Center problem is $\text{W}[1]$-hard for the combined parameter $(k, p, h)$, where $p$ is the pathwidth and $h$ the highway dimension according to Definition 3.1 for any universal constant $c \geq 4$. Moreover, there is no $f(k, p, h) \cdot n^{o(p + \sqrt{k + h})}$ time algorithm$^8$ for the same restriction on the input graphs, for any computable function $f$, under ETH.

Note that in this theorem we also add the pathwidth as a parameter, which arguably is not very useful to model transportation networks, since road networks of large cities (especially on the American continent) can contain large grids, which implies that the pathwidth will be rather large. We include this well-studied parameter here nonetheless, since by a result of Katsikarelis et al. [KLP19] it is known that an EPAS exists when parameterizing by the pathwidth (or even the tree- or cliquewidth). Thus Theorem 3.16 complements not only this result, but also the above ones for planar, low doubling, and low highway dimension graphs, by showing that approximations are necessary in each case. But furthermore, even if one were to combine all the models presented in Section 3.1 and assume that a transportation network is planar, is embeddable into some metric of constant doubling dimension, has bounded highway dimension, and even has bounded pathwidth, the $k$-Center problem cannot be solved efficiently, unless FPT=\text{W}[1]. Thus it seems unavoidable to approximate the problem in transportation networks when developing fast algorithms. A recent result by Blum [Blu20] also shows that the skeleton dimension, which is yet another graph parameter for transportation networks, can be added to the list of parameters in Theorem 3.16 while still obtaining $\text{W}[1]$-hardness.

To obtain Theorem 3.16, in [11] we give a reduction from the Grid Tiling with Inequality problem, which was introduced by Marx and Sidiropoulos [MS14] and is defined as follows. Given

---

$^8$Here $o(p + \sqrt{k + h})$ means $g(p + \sqrt{k + h})$ for any function $g$ such that $g(x) \in o(x)$. 

28
$\kappa^2$ non-empty sets $S_{i,j} \subseteq [n]^2$ of pairs of integers, where $i, j \in [\kappa]$, the task is to select one pair $s_{i,j} \in S_{i,j}$ for each set such that

- if $s_{i,j} = (a, b)$ and $s_{i+1,j} = (a', b')$ for $i \leq \kappa - 1$ then $a \leq a'$, and
- if $s_{i,j} = (a, b)$ and $s_{i,j+1} = (a', b')$ for $j \leq \kappa - 1$ then $b \leq b'$.

The Grid Tiling with Inequality problem is $\text{W}[1]$-hard for parameter $\kappa$, and moreover, under ETH has no $f(\kappa) \cdot n^{o(\kappa)}$ time algorithm for any computable function $f$.

This problem is typically used to show hardness of various problems on planar graphs or $\ell_q$-metrics in the plane. On a high level, we can think of each set $S_{i,j}$ of an instance of Grid Tiling with Inequality as the contents of a cell with coordinates $(i, j)$ in a $\kappa \times \kappa$ grid. For a reduction, we may introduce a gadget for each cell, which encodes the integer pairs of the corresponding set $S_{i,j}$, and the gadgets are then arranged in a grid-like fashion. If each gadget is a planar graph then so is the overall constructed graph. For $\ell_q$-metrics in the plane, each gadget is given by a point set contained in a square corresponding to a cell of the grid. In our reduction for Theorem 3.16, ideally we would like to combine both these approaches in order to obtain a planar graph of constant doubling dimension. Thus our gadgets are edge-weighted planar graphs, but it is not clear whether the vertices can be mapped to points in the plane so that for some $q$ the $\ell_q$-distances are the same as distances in the resulting planar graph. Instead of mapping the constructed graph into the plane, we exploit the grid-like structure of the graph and use the intuition that the distances of a regular grid in the plane abide to the $\ell_1$-norm. In particular, we show that every ball in the graph can be covered by at most 324 balls of half the radius, and thus the doubling dimension is constant.

We then go on to bound all parameters in our reduction for Theorem 3.16 in terms of $\kappa$. For $k$ we show that an instance of Grid Tiling with Inequality has a solution if and only if the constructed graph has a $k$-Center solution of a certain bounded cost, in which each gadget needs exactly 5 centers, i.e., $k = 5\kappa^2$. To bound the pathwidth $p$ and the highway dimension $h$ in our reduction we again use the grid structure of the constructed graph. In particular, it is well-known that a $\kappa \times \kappa$ grid graph has pathwidth $\Theta(\kappa)$, and we are able to modify a corresponding path decomposition so that each bag only contains a constant number of additional vertices to incorporate the gadgets. For the highway dimension, Abraham et al. [Abr+10] note that in a regular $\kappa \times \kappa$ grid graph a ball of radius $\Theta(\kappa)$ contains $\Theta(\kappa)$ vertical (and horizontal) vertex disjoint paths of length $\Omega(\kappa)$, each of which will need to be hit by a hub. Thus the highway dimension of a $\kappa \times \kappa$ grid graph is $\Omega(\kappa)$. In our constructed graph for the reduction, two gadgets of neighbouring cells of the $\kappa \times \kappa$ grid are connected by only one path. Therefore we can bound the highway dimension from above by placing a hub on each of these connecting paths (i.e., between grid cells), in order to hit long vertical and horizontal paths. Additionally, we are able to place hubs in each gadget such that any ball of a given radius contains only a constant number of them. As there are $\Theta(\kappa^2)$ points where gadgets connect, this implies a highway dimension of $O(\kappa^2)$ for the constructed graph.
References to related work

dimension and provably efficient shortest path algorithms’. *Journal of the ACM*
63.5 (2016), p. 41.

and shortest path algorithms’. In: *Proceedings of the International Colloquium on

shortest paths, and provably efficient algorithms’. In: *Proceedings of the ACM-

[AP02] P. K. Agarwal and C. M. Procopiuc. ‘Exact and approximation algorithms for

algorithms for dominating set and related problems on planar graphs’. *Algorithmica*

[Aro98] S. Arora. ‘Polynomial time approximation schemes for Euclidean traveling salesman

medians and Related Problems’. In: *Proceedings of the ACM Symposium on Theory

Approximation of Euclidean k-Means’. In: *Proceedings of the 31st International


[BKS18] A. Becker, P. N. Klein and D. Saulpic. ‘Polynomial-Time Approximation Schemes
for k-Center, k-Median, and Capacitated Vehicle Routing in Bounded Highway

Computation (IPEC)*. 2019, 4:1–4:15.

[Blu20] J. Blum. ‘W[1]-Hardness of the k-Center Problem Parameterized by the Skeleton Di-
mension’. In: *In Proceedings of the 26th International Computing and Combinatorics
Conference (COCOON)*. Accepted for publication. 2020.

exponential time algorithms for connectivity problems parameterized by treewidth’.

[BD97] A. Borchers and D.-Z. Du. ‘The k-Steiner Ratio in Graphs’. *SIAM journal on


31


Appendix: Papers Summarized in This Thesis
FPT Inapproximability of Directed Cut and Connectivity Problems

Rajesh Chitnis
School of Computer Science, University of Birmingham, UK
rajeshchitnis@gmail.com

Andreas Emil Feldmann
Charles University, Czechia
feldmann.a.e@gmail.com

Abstract

Cut problems and connectivity problems on digraphs are two well-studied classes of problems from the viewpoint of parameterized complexity. After a series of papers over the last decade, we now have (almost) tight bounds for the running time of several standard variants of these problems parameterized by two parameters: the number \( k \) of terminals and the size \( p \) of the solution. When there is evidence of FPT intractability, then the next natural alternative is to consider FPT approximations.

In this paper, we show two types of results for directed cut and connectivity problems, building on existing results from the literature: first is to circumvent the hardness results for these problems by designing FPT approximation algorithms, or alternatively strengthen the existing hardness results by creating “gap-instances” under stronger hypotheses such as the (Gap-)Exponential Time Hypothesis (ETH). Formally, we show the following results:

**Cutting paths between a set of terminal pairs, i.e., Directed Multicut:**

- Directed Multicut has a \( k/2 \)-approximation in \( 2^{O(p^2)} \cdot n^{O(1)} \) time (i.e., a 2-approximation if \( k = 4 \)),

- Under Gap-ETH, Directed Multicut does not admit an \( (\frac{k}{2} - \epsilon) \)-approximation in \( f(p) \cdot n^{O(1)} \) time, for any computable function \( f \), even if \( k = 4 \).

**Pairwise connecting a set of terminals, i.e., Strongly Connected Steiner Subgraph (SCSS):**

- SCSS remains W[1]-hard parameterized by \( p + k \) due to Guo et al. [SIDMA '11]. Dinur and Manurangsi [ITCS '18] further showed that there is no FPT \( k^{1/4 - o(1)} \)-approximation algorithm parameterized by \( k \), under Gap-ETH.

- SCSS is W[1]-hard parameterized by \( k + p \). Moreover, under ETH, there is no \( (1 + \epsilon) \)-approximation for SCSS in \( f(k, p, \epsilon) \cdot n^{o(k + \sqrt{k + p + \frac{7}{10}})} \) time for any computable function \( f \).

- SCSS has no \( (2 - \epsilon) \)-approximation in FPT time parameterized by \( k \), under Gap-ETH. This answers in the negative a question of Chitnis et al. [ESA '18].

- SCSS is W[1]-hard parameterized by \( k + p \). Moreover, under ETH, there is no \( (1 + \epsilon) \)-approximation for SCSS in \( f(k, p, \epsilon) \cdot n^{o(k + \sqrt{k + p + \frac{7}{10}})} \) time for any computable function \( f \).

Previously, the only known FPT approximation results for SCSS applied to general graphs parameterized by \( k \): a 2-approximation by Chitnis et al. [IPEC '13], and a matching \( (2 - \epsilon) \)-hardness under Gap-ETH by Chitnis et al. [ESA '18].
1 Introduction

Given a weighted directed graph \( G = (V, E) \) with two terminal vertices \( s, t \) the problems of finding a minimum weight \( s \rightarrow t \) cut and a minimum weight \( s \rightarrow t \) path can both be famously solved in polynomial time. There are two natural generalizations when we consider more than two terminals: either we look for connectivity/cuts between all terminals of a given set, or we look for connectivity/cuts between a given set of terminal pairs. This leads to the four problems of Directed Multiway Cut, Directed Multicut, Strongly Connected Steiner Subgraph and Directed Steiner Network:

- **Cutting all paths between a set of terminals**: In the Directed Multiway Cut problem, we are given a set of terminals \( T = \{t_1, t_2, \ldots, t_k\} \) and the goal is to find a minimum weight subset \( X \subseteq V \) such that \( G \setminus X \) has no \( t_i \rightarrow t_j \) path for any \( 1 \leq i \neq j \leq k \).

- **Cutting paths between a set of terminal pairs**: In the Directed Multicut problem, we are given a set of terminal pairs \( T = \{(s_i, t_i)\}_{i=1}^k \) and the goal is to find a minimum weight subset \( X \subseteq V \) such that \( G \setminus X \) has no \( s_i \rightarrow t_i \) path for any \( 1 \leq i \leq k \).

- **Connecting all terminals of a given set**: In the Strongly Connected Steiner Subgraph (SCSS) problem, we are given a set of terminals \( T = \{t_1, t_2, \ldots, t_k\} \) and the goal is to find a minimum weight subset \( X \subseteq V \) such that \( G[X] \) has a \( t_i \rightarrow t_j \) path for every \( 1 \leq i \neq j \leq k \).

- **Connecting a set of terminal pairs**: In the Directed Steiner Network (DSN) problem, we are given a set of terminal pairs \( T = \{(s_i, t_i)\}_{i=1}^k \) and the goal is to find a minimum weight subset \( X \subseteq V \) such that \( G[X] \) has an \( s_i \rightarrow t_i \) path for every \( 1 \leq i \leq k \).

All four of the aforementioned problems are known to be NP-hard, even for small values of \( k \). One way to cope with NP-hardness is to try to design polynomial time approximation algorithms with small approximation ratio. However, apart from Directed Multiway Cut, which admits a 2-approximation in polynomial time \([35]\), all the other three problems are known to have strong lower bounds (functions of \( n \)) on the approximation ratio of polynomial time algorithms \([16, 19, 25]\). Another way to cope with NP-hardness is to try to design FPT algorithms. However, apart from Directed Multiway Cut which has an FPT algorithm parameterized by the size \( p \) of the cutset, all the other three problems are known to be \( W[1] \)-hard (and hence fixed-parameter intractable) parameterized by size \( p \) of the solution \( X \) plus the number \( k \) of terminals/terminal pairs. When neither of the paradigms of polynomial time approximation algorithms nor (exact) FPT algorithm seem to be successful, the next natural alternative is to try to design FPT approximation algorithms or show hardness of FPT approximation results.
In this paper, we consider the remaining three problems of Directed Multicut, Strongly Connected Steiner Subgraph and Directed Steiner Network, for which strong approximation and parameterized lower bounds exist, from the viewpoint of FPT approximation algorithms. We obtain two types of results for these three problems: the first is to circumvent the W[1]-hardness and polynomial-time inapproximability results for these problems by designing FPT approximation algorithms, and the second is to strengthen the existing W[1]-hardness by creating “gap-instances” under stronger hypotheses than FPT ≠ W[1] such as (Gap-) Exponential Time Hypothesis (ETH). Throughout, we use $k$ to denote number of terminals or terminal pairs and $p$ to denote size of the solution. First, in Section 1.1, we give a brief overview of the current state-of-the-art results for each the three problems from the lens of polynomial time approximation algorithms, FPT algorithms, and FPT approximation algorithms followed by the formal statements of our results. Then, in Section 1.2 we describe the recent flux of results which have set up the framework of FPT hardness of approximation under (Gap-)ETH, and how we use it obtain our hardness results in this paper.

1.1 Previous work and our results

The Directed Multicut problem

Garg et al. [23] showed that Directed Multicut is NP-hard even for $k = 2$. The current best approximation ratio in terms of $n$ is $O(n^{11/23} \cdot \log^{O(1)} n)$ due to Agarwal et al. [1], and it is known that Directed Multicut is hard to approximate in polynomial time to within a factor of $2^{O((\log n) \cdot \epsilon)}$ for any constant $\epsilon > 0$, unless NP ⊆ ZPP [16]. There is a simple $k$-approximation in polynomial time obtained by solving each terminal pair as a separate instance of min $s \rightarrow t$ cut and then taking the union of all the $k$ cuts. Chekuri and Madan [8] and later Lee [30] showed that this is tight: assuming the Unique Games Conjecture of Khot [28], it is not possible to approximate Directed Multicut better than factor $k$ in polynomial time, for any fixed $k$. On the FPT side, Marx and Razgon [34] showed that Directed Multicut is W[1]-hard paramterized by $p$. For the case of bounded $k$, Chitnis et al. [14] showed that Directed Multicut is FPT parameterized by $p$ when $k = 2$, but Pilipczuk and Wahlstrom [36] showed that the problem remains W[1]-hard parameterized by $p$ when $k = 4$. The status of Directed Multicut parameterized by $p$ when $k = 3$ is an outstanding open question. We first obtain the following FPT approximation for Directed Multicut parameterized by $p$, which beats any approximation obtainable when parameterizing by $k$ (even in XP time) according to [8, 30]:

\begin{itemize}
  \item \textbf{Theorem 1.} The Directed Multicut problem admits an $\lceil k/2 \rceil$-approximation in $2^{O(p^3)} \cdot n^{O(1)}$ time.
\end{itemize}

The proof of the above theorem uses the FPT algorithm of Chitnis et al. [14, 12] for Directed Multiway Cut parameterized by $p$ as a subroutine. Note that Theorem 1 gives an FPT 2-approximation for Directed Multicut With 4 Pairs. We complement this upper bound with a constant factor lower bound for approximation ratio of any FPT algorithm for Directed Multicut With 4 Pairs.

\begin{itemize}
  \item \textbf{Theorem 2.} Under Gap-ETH, for any $\epsilon > 0$ and any computable function $f$, there is no $f(p) \cdot n^{O(1)}$ time algorithm that computes an $(\frac{59}{58} - \epsilon)$-approximation for Directed Multicut With 4 Pairs.
\end{itemize}

We did not optimize the constant $59/58$ in order to keep the analysis simple: we believe it can be easily improved, but our techniques would not take it close to the upper bound of 2.
FPT Inapproximability of Directed Cut and Connectivity Problems

The Directed Steiner Network (DSN) problem

The DSN problem is known to be NP-hard, and furthermore even computing an $O(2^{\log^{1-\varepsilon} n})$-approximation is not possible [19] in polynomial time, unless NP $\subseteq$ DTIME$(n^{\text{polylog}(n)})$. The best known approximation factors for polynomial time algorithms are $O(n^{1/2+\varepsilon})$ and $O(k^{1/2+\varepsilon})$ [4, 7, 21]. On the FPT side, Feldman and Ruhl [20] designed an $n^{O(k)}$ algorithm for DSN (cf. [22]). Chitnis et al. [15] showed that the Feldman-Ruhl algorithm is tight: under ETH, there is no $f(k) \cdot n^{o(k)}$ algorithm (for any computable function $f$) for DSN even if the input graph is a planar directed acyclic graph. Guo et al. [24] showed that DSN remains W[1]-hard even when parameterized by the larger parameter $k+p$. Dinur and Manurangsi [18] further showed that DSN on general graphs has no FPT approximation algorithm with ratio $k^{1/4-\alpha(1)}$ when parameterized by $k$, under Gap-ETH.

Chitnis et al. [11] considered two relaxations of the Directed Steiner Network problem: the bi-DSN problem where the input graph is bidirected, and the DSN\textsubscript{Planar} problem where the input graph is general but the goal is to find a solution whose cost is at most that of an optimal planar solution (if one exists). The main result of Chitnis et al. [11] is that although bi-DSN\textsubscript{Planar} (i.e., the intersection of bi-DSN and DSN\textsubscript{Planar}) is W[1]-hard parameterized by $k+p$, it admits a parameterized approximation scheme: for any $\varepsilon > 0$, there is a $\max\{2^{k^{O(1/\varepsilon)}}, n^{2^{O(1/\varepsilon)}}\}$ time algorithm for bi-DSN\textsubscript{Planar} which computes a $(1+\varepsilon)$-approximation. Such a parameterized approximation is not possible for bi-DSN as Chitnis et al. [11] showed that under Gap-ETH there is a constant $\alpha > 0$ such that there is no FPT $\alpha$-approximation. They asked whether a parameterized approximation scheme for the remaining variant of DSN, i.e., the DSN\textsubscript{Planar} problem, exists. We answer this question in the negative with the following lower bound.

Theorem 3. Under Gap-ETH, for any $\varepsilon > 0$ and any computable function $f$, there is no $f(k) \cdot n^{O(1)}$ time algorithm that computes a $(2-\varepsilon)$-approximation for DSN\textsubscript{Planar}, even if the input graph is a directed acyclic graph (DAG).

The W[1]-hardness proof of [15] for DSN on planar graphs parameterized by $k$ does not give hardness parameterized by $p$ since in that reduction the value of $p$ grows with $n$. Our next result shows that the slightly more general problem of DSN\textsubscript{Planar} (here the input graph is general, but we want to find a solution of cost $\leq p$ if there is a planar solution of size $\leq p$) is indeed W[1]-hard parameterized by $k+p$. Also we obtain a lower bound for approximation schemes for this problem under ETH, i.e., under a weaker assumption than the one used for Theorem 3.

Theorem 4. The DSN\textsubscript{Planar} problem is W[1]-hard parameterized by $p+k$, even if the input graph is a directed acyclic graph (DAG). Moreover, under ETH, for any computable function $f$

1. there is no $f(k,p) \cdot n^{o(k+p)}$ time algorithm for DSN\textsubscript{Planar}, and
2. there is no $f(k,\varepsilon, p) \cdot n^{o(k+p+\varepsilon/2)}$ time algorithm which computes a $(1+\varepsilon)$-approximation for DSN\textsubscript{Planar} for every $\varepsilon > 0$.

Note that just the W[1]-hardness of DSN\textsubscript{Planar} parameterized by $k+p$ already follows from [11] who showed that even the special case of bi-DSN\textsubscript{Planar} is W[1]-hard parameterized by $k+p$. However, this reduction from [11] was from $\ell$-Clique to an instance of bi-DSN\textsubscript{Planar} with $k = O(\ell^2)$ and $p = O(\ell^2)$, whereas Theorem 4 gives a reduction from $\ell$-Clique to DSN\textsubscript{Planar} with $k = O(\ell)$ and $p = O(\ell^2)$. This gives much improved lower bounds on the running times.

---

1 Bidirected graphs are directed graphs which have the property that for every edge $u \to v$ in $G$ the reverse edge $v \to u$ exists in $G$ as well and moreover has the same weight as $u \to v$.

2 In the following, $o(f(k,p,\varepsilon))$ means any function $g(f(k,p,\varepsilon))$ such that $g(x) \in o(x)$. 
The Strongly Connected Steiner Subgraph (SCSS) problem

The SCSS problem is NP-hard, and the best known approximation ratio in polynomial time for SCSS is $k^\epsilon$ for any $\epsilon > 0$ [6]. A result of Halperin and Krauthgamer [25] implies SCSS has no $\Omega(\log^{-1-\epsilon} n)$-approximation for any $\epsilon > 0$, unless NP has quasi-polynomial Las Vegas algorithms. On the FPT side, Feldman and Ruhl [20] designed an $n^{O(k)}$ algorithm for SCSS (cf. [22]). Chitnis et al. [15] showed that the Feldman-Ruhl algorithm is almost optimal: under ETH, there is no $f(k) \cdot n^{o(k/\log k)}$ algorithm (for any computable function $f$) for SCSS. Guo et al. [24] showed that SCSS remains W[1]-hard even when parameterized by the larger parameter $k + p$. Chitnis et al. [11] showed that the SCSS problem restricted to bidirected graphs remains NP-hard, but is FPT parameterized by $k$. The SCSS problem admits a square-root phenomenon on planar graphs: Chitnis et al. [15] showed that SCSS on planar graphs has an $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ algorithm, and under ETH there is a tight lower bound of $f(k) \cdot n^{o(\sqrt{k})}$ for any computable function $f$. The W[1]-hardness proof of [15] for SCSS on planar graphs parameterized by $k$ does not give hardness parameterized by $p$, since in that reduction the value of $p$ grows with $n$. Our next result shows that the slightly more general problem of SCSS\textsubscript{PLANAR} (here the input graph is general, but we want to find a solution of cost $\leq p$ if there is a planar solution of size $\leq p$) is indeed W[1]-hard parameterized by $k + p$. We also obtain a lower bound for approximation schemes for this problem under ETH:

**Theorem 5.** The SCSS\textsubscript{PLANAR} problem is W[1]-hard parameterized by $p + k$. Moreover, under ETH, for any computable function $f$

- there is no $f(k, p) \cdot n^{o(\sqrt{k + p})}$ time algorithm for SCSS\textsubscript{PLANAR}, and
- there is no $f(k, \epsilon, p) \cdot n^{o(\sqrt{k + p + 1/\epsilon})}$ time algorithm which computes an $(1 + \epsilon)$-approximation for SCSS\textsubscript{PLANAR} for every $\epsilon > 0$.

To the best of our knowledge, the only known FPT approximation results for SCSS applied to general graphs parameterized by $k$: a simple FPT 2-approximation due to Chitnis et al. [13], and a matching $(2 - \epsilon)$-hardness (for any constant $\epsilon > 0$) under Gap-ETH due to Chitnis et al. [11].

### 1.2 FPT inapproximability results under (Gap-)ETH

A standard hypothesis for showing lower bounds for running times of FPT and exact exponential time algorithms is the Exponential Time Hypothesis (ETH) of Impagliazzo and Paturi [26].

**Hypothesis 6.** Exponential Time Hypothesis (ETH): There exists a constant $\delta > 0$ such that no algorithm can decide whether any given 3-CNF formula is satisfiable in time $O(2^{\delta m})$ where $m$ denotes the number of clauses.

The original conjecture stated the lower bound as exponential in terms of the number of variables not clauses, but the above statement follows from the Sparsification Lemma of [27]. The Exponential Time Hypothesis has been used extensively to show a variety of lower bounds including those for FPT algorithms, exact exponential time algorithms, hardness of polynomial time approximation, and hardness of FPT approximation. We refer the interested reader to [31] for a survey on lower bounds based on ETH.

To show the W[1]-hardness of DSN\textsubscript{PLANAR} (Theorem 4) and SCSS\textsubscript{PLANAR} (Theorem 5) parameterized by $k + p$ we design parameterized reductions from $\ell$-Clique to these problems such that $\max\{k, p\}$ is upper bounded by a function of $\ell$. Furthermore, by choosing $\epsilon$ to be small enough such that computing an $(1 + \epsilon)$-approximation is the same as computing the
optimal solution, we also obtain runtime lower bounds for \((1 + \epsilon)\)-approximations for these two problems by translating the \(f(\ell) \cdot n^o(\ell)\) lower bound for \(\ell\)-Clique \([9]\) under ETH (for any computable function \(f\)).

Recently, a gap version of the ETH was proposed:

**Hypothesis 7. Gap-ETH \([17, 32]\):** There exists a constant \(\delta > 0\) such that, given a 3CNF formula \(\Phi\) on \(n\) variables, no \(2^{o(n)}\)-time algorithm can distinguish between the following two cases correctly with probability at least 2/3:

- \(\Phi\) is satisfiable.
- Every assignment to the variables violates at least a \(\delta\)-fraction of the clauses of \(\Phi\).

It is known \([5, 2]\) that Gap-ETH follows from ETH given other standard conjectures, such as the existence of linear sized PCPs or exponentially-hard locally-computable one-way functions. We refer the interested reader to \([17, 5]\) for a discussion on why Gap-ETH is a plausible assumption. In a breakthrough result, Chalermsook et al. \([5]\) used Gap-ETH to show that the two famous parameterized intractable problems of Clique and Set Cover are completely inapproximable in FPT time parameterized by the size of the solution. In this paper, we obtain two hardness of approximation results (Theorem 2 and Theorem 3) based on Gap-ETH. The starting point of our hardness of approximation results are based on the recent results on parameterized inapproximability of the Densest \(k\)-Subgraph problem. Recall that, in the Densest \(k\)-Subgraph (D\&S) problem \([29]\), we are given an undirected graph \(G = (V, E)\) and an integer \(k\) and the goal is to find a subset \(S \subseteq V\) of size \(\ell\) that induces as many edges in \(G\) as possible. Chalermsook et al. \([5]\) showed that, under randomized Gap-ETH, there is no FPT approximation (parameterized by \(k\)) with ratio \(k^{o(1)}\).

This was improved recently by Dinur and Manurangsi \([18]\) who showed better hardness and under deterministic Gap-ETH. We state their result formally\(^3\):

**Theorem 8 (\([18, \text{Theorem 2}]\)).** Under Gap-ETH, for any function \(h(\ell) = o(1)\), there is no \(f(\ell) \cdot n^{O(1)}\)-time algorithm that, given a graph \(G\) on \(n\) vertices and an integer \(k\), can distinguish between the following two cases:

- (YES) \(G\) contains at least one \(\ell\)-clique as a subgraph.
- (NO) Every \(\ell\)-subgraph of \(G\) contains less than \(\ell^{h(\ell) - 1} \cdot (\frac{\ell}{2})^2\) edges.

Note that this result is essentially tight: there is a simple \(O(\ell)\) approximation since the number of edges induced by an \(\ell\)-vertex subgraph is at most \(\binom{\ell}{2}\) and at least \([\ell/2]\) (without loss of generality, we can assume there are no isolated vertices). Instead of working with D\&S, we will reduce from a “colored” version of the problem called **Maximum Colored Subgraph Isomorphism**, which can be defined as follows.

<table>
<thead>
<tr>
<th>Maximum Colored Subgraph Isomorphism (MCSI)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong>: An instance (\Gamma) of MCSI consists of three components:</td>
</tr>
<tr>
<td>- An undirected graph (G = (V_G, E_G)),</td>
</tr>
<tr>
<td>- A partition of vertex set (V_G) into disjoint subsets (V_1, \ldots, V_\ell),</td>
</tr>
<tr>
<td>- An undirected graph (H = (V_H = {1, \ldots, \ell}, E_H)).</td>
</tr>
<tr>
<td><strong>Goal</strong>: Find an assignment (\phi : V_H \to V_G) where (\phi(i) \in V_i) for every (i \in [\ell]) that maximizes the number of edges (i - j \in E_H) such that (\phi(i) - \phi(j) \in E_G).</td>
</tr>
</tbody>
</table>

---

\(^3\) Dinur and Manurangsi \([18]\) actually state their result for 2-CSPs.
This problem is referred to as Label Cover in the hardness of approximation literature [3]. However, Chitnis et al. [11] used the name Maximum Colored Subgraph Isomorphism to be consistent with the naming conventions in the FPT community: this problem is an optimization version of Colored Subgraph Isomorphism [33]. The graph \( H \) is sometimes referred to as the supergraph of \( \Gamma \). Similarly, the vertices and edges of \( H \) are called supernodes and superedges of \( \Gamma \). Moreover, the size of \( \Gamma \) is defined as \( n = |V_G| \), the number of vertices of \( G \). Additionally, for each assignment \( \phi \), we define its value \( \text{val}(\phi) \) to be the fraction of superedges \( i - j \in E_H \) such that \( \phi(i) - \phi(j) \in E_G \); such superedges are said to be covered by \( \phi \). The objective of MCSI is now to find an assignment \( \phi \) with maximum value. We denote the value of the optimal assignment by \( \text{val}(\Gamma) \), i.e., \( \text{val}(\Gamma) = \max_\phi \text{val}(\phi) \).

Using Theorem 8 we derive the following two corollaries regarding hardness of approximation for Maximum Colored Subgraph Isomorphism when the supergraph \( H \) has special structure. These corollaries follow quite straightforwardly from Theorem 8 using the idea of splitters, but we provide proofs in the full version [10] for completeness.

\[ \text{Corollary 9.} \quad [\star] \quad \text{Assuming Gap-ETH, for any function } h(\ell) = o(1), \text{ there is no } f(\ell) \cdot n^{O(1)} \text{-time algorithm that, given a MCSI instance } \Gamma \text{ of size } n \text{ such that the supergraph } H = K_\ell, \text{ can distinguish between the following two cases:} \]

\[ \begin{align*}
(\text{YES}) & \quad \text{val}(\Gamma) = 1. \\
(\text{NO}) & \quad \text{val}(\Gamma) < \ell^{h(\ell) - 1}.
\end{align*} \]

\[ \text{Corollary 10.} \quad [\star] \quad \text{Assuming Gap-ETH, for any function } h(\ell) = o(1), \text{ there is no } f(\ell) \cdot n^{O(1)} \text{-time algorithm that, given a MCSI instance } \Gamma \text{ of size } n \text{ such that the supergraph } H \text{ is the complete bipartite subgraph } K_{\ell^2, \ell^2}, \text{ can distinguish between the following two cases:} \]

\[ \begin{align*}
(\text{YES}) & \quad \text{val}(\Gamma) = 1. \\
(\text{NO}) & \quad \text{val}(\Gamma) < \ell^{h(\ell) - 1}.
\end{align*} \]

We prove Theorem 2 and Theorem 3 via reductions from Corollary 9 and Corollary 10 respectively.

## 2 FPT (In)Approximability of Directed Multicut

In this section we design an FPT 2-approximation for Directed Multicut With 4 Pairs parameterized by \( p \) (Section 2.1) and complement this with a lower bound (Section 2.2) showing that no FPT algorithm (parameterized by \( p \)) for Directed Multicut With 4 Pairs can achieve a ratio of \( \left( \frac{59}{58} - \epsilon \right) \) under Gap-ETH.

### 2.1 FPT approximation algorithm

It is well-known that a \( k \)-approximation can be computed in polynomial time by taking union of min cuts of each of the \( k \) terminal pairs. Chekuri and Madan [8] and later Lee [30] showed that this approximation ratio is best-possible for polynomial time algorithms under the Unique Games Conjecture of Khot [28]. The same lower bound also applies for any constant \( k \), i.e., even an XP algorithm parameterized by \( k \) cannot compute a better approximation than a polynomial time algorithm. We now design an FPT \( \lceil k/2 \rceil \)-approximation for Directed Multicut. The idea is borrowed from the proof of Chitnis et al. [14] that Directed Multicut With 2 Pairs is FPT parameterized by \( p \).

\[ \text{All proofs labelled with } [\star] \text{ appear in the full version [10]} \]
Theorem 1. The Directed Multicut problem admits a \([k/2]\)-approximation in \(2^{O(p^2)} \cdot n^{O(1)}\) time. Formally, the algorithm takes an instance \((G, T)\) of Directed Multicut and in \(2^{O(p^2)} \cdot n^{O(1)}\) time either concludes that there is no solution of cost at most \(p\), or produces a solution of cost at most \(p[k/2]\).

Proof. Let the pairs be \(T = \{(s_i, t_i) : 1 \leq i \leq k\}\), and let \(OPT\) be the optimum value for the instance \((G, T)\) of Directed Multicut. For now, assume that \(k\) is even. Introduce \(k/2\) new vertices \(r_j, q_j\), for \(1 \leq j \leq k/2\), of weight \(p + 1\) each, and add the following edges:

- \(r_j \rightarrow s_{2j-1}\) and \(t_{2j-1} \rightarrow q_j\)
- \(q_j \rightarrow s_{2j}\) and \(t_{2j} \rightarrow r_j\)

Let the resulting graph be \(G'\), and note that \(G\) has an \(s_i \rightarrow t_i\) path for some \(1 \leq i \leq k\) if and only if \(G'\) has a \(q_i/2 \rightarrow r_i/2\) or \(r_{i-1}/2 \rightarrow q_{i-1}/2\) path (depending on whether \(i\) is even or odd). Since the vertices \(r_j, q_j\) have weight \(p + 1\) each, it follows that \(G\) has a solution of size at most \(p\) for the instance \((G, \{(s_{2j-1}, t_{2j-1}), (s_{2j}, t_{2j})\})\) of Directed Multicut if and only if \(G'\) has a solution of size at most \(p\) for the Directed Multiway Cut instance with input graph \(G\) and terminals \(r_j, q_j\). We use the algorithm of Chitnis et al. [14, 12] for Directed Multiway Cut which checks in \(2^{O(p^2)} \cdot n^{O(1)}\) time if there is a solution of cost at most \(p\). If there is no solution of cost at most \(p\) between \(r_j, q_j\) in \(G'\) then this implies that \(G\) has no cut of size at most \(p\) separating \((s_{2j-1}, t_{2j-1})\) and \((s_{2j}, t_{2j})\) and hence \(OPT > p\). Otherwise, there is a cut \(C_j\) in \(G\) of cost at most \(p\) which separates \((s_{2j-1}, t_{2j-1})\) and \((s_{2j}, t_{2j})\).

The output of the algorithm is the cut \(C = \bigcup_{j=1}^{k/2} C_j\). Clearly, if \(k\) is even then \(C\) is a feasible solution for the instance \((G, T)\) of Directed Multicut with cost at most \(\sum_{j=1}^{k/2} \text{cost}(C_j) \leq pk/2\). In case \(k\) is odd we use the above procedure for the terminal pairs \(\{(s_i, t_i) : 1 \leq i \leq k-1\}\), and finally add a min cut between the last terminal pair \((s_k, t_k)\). This results in the desired \([k/2]\)-approximation.

2.2 No FPT \((\frac{59}{58} - \epsilon)\)-approximation under Gap-ETH

With the parameterized hardness of approximating MCSI ready, we can now prove our hardness results for Directed Multicut with 4 terminal pairs.

Theorem 2. Under Gap-ETH, for any \(\epsilon > 0\) and any computable function \(f\), there is no \(f(p) \cdot n^{O(1)}\) time algorithm that computes an \((\frac{59}{58} - \epsilon)\)-approximation for Directed Multicut With 4 Pairs.

Our proof of the parameterized inapproximability of Directed Multicut With 4 Pairs is based on a reduction from Maximum Colored Subgraph Isomorphism whose properties are described below.

Lemma 11. There exists a polynomial time reduction that, given an instance \(\Gamma = (G, K, V_1 \cup \cdots \cup V_t)\) of MCSI, produces an instance \((G', T')\) of Directed Multicut With 4 Pairs such that

- (Completeness): If \(\text{val}(\Gamma) = 1\), then there exists a solution \(N \subseteq V(G')\) of cost \(29\ell^2\) for the instance \((G', T')\) of Directed Multicut With 4 Pairs
- (Soundness): If \(\text{val}(\Gamma) < \frac{1}{10}\), then every solution \(N \subseteq V(G')\) for the instance \((G', T')\) of Directed Multicut With 4 Pairs has cost more than \(29.5\ell^2\).
- (Parameter Dependency): The size of the solution is \(p = O(\ell^2)\).

\(^5\) This is independent of number of the terminals
In the proof of Lemma 11, we actually use the same reduction as from [36], but with different weights. We reduce to the vertex-weighted variant of Directed Multicut With 4 Pairs where we have four different types of weights for the vertices:

- light vertices (shown using gray color) which have weight $B = \frac{\ell^2}{(2\ell)^2}$
- medium vertices (shown using green color) which have weight $2B$
- heavy vertices (shown using orange color) which have weight $20\ell$
- super-heavy vertices (shown using white color) which have weight $100\ell^2$

2.2.1 Construction of the Directed Multicut With 4 Pairs instance

Without loss of generality (by adding isolated vertices if necessary) we can assume that $|V_i| = n$ for each $i \in [\ell]$. For each $i \in [\ell]$ let $V_i = \{v_i, v_{i2}, \ldots, v_{i\ell}\}$. Then $|V(G)| = n\ell$. We now describe the construction of the (vertex-weighted) Directed Multicut With 4 Pairs instance $(G', T')$.

- Introduce eight terminals, arranged in four terminal pairs as follows:
  \[ T' = \{(s^0_{0-n}, s^1_{0-n}), (s^0_{n-0}, s^0_{n-0}), (s^1_{n-0}, t^0_{n-0}), (s^1_{n-0}, t^1_{n-0})\} \]
  Each of the 8 terminals is super-heavy.

- For every $1 \leq i \leq \ell$, we introduce a bidirected path on $2n + 1$ vertices (see Figure 2)
  \[ Z_i := z_0^i \leftrightarrow z_1^i \leftrightarrow z_2^i \leftrightarrow \ldots \leftrightarrow z_n^i \leftrightarrow z_n^i, \]
  called henceforth the z-path for color class $i$. For each $0 \leq a \leq n$ the vertex $z_a^i$ is super-heavy and for each $1 \leq a \leq n$ the vertex $z_a^i$ is heavy.

- For every pair $(i, j)$ where $1 \leq i, j \leq \ell$, $i \neq j$, we introduce two bidirected paths (see Figure 2 and Figure 1) on $2n + 1$ vertices
  \[ X_{i,j} := x_{i0}^j \leftrightarrow x_{i1}^j \leftrightarrow x_{i2}^j \leftrightarrow \ldots \leftrightarrow x_{in}^j \leftrightarrow x_{in}^j \]
  and
  \[ Y_{i,j} := y_{i0}^j \leftrightarrow y_{i1}^j \leftrightarrow y_{i2}^j \leftrightarrow \ldots \leftrightarrow y_{in}^j \leftrightarrow y_{in}^j \]
  We call these paths the x-path and the y-path for the pair $(i, j)$. For each $0 \leq a \leq n$ the vertices $x_a^i, j$ and $y_a^i, j$ are super-heavy. For each $1 \leq a \leq n$ the vertices $x_a^i, j$ and $y_a^i, j$ are medium.

- For every pair $(i, j)$ with $1 \leq i, j \leq \ell$, $i \neq j$, and every $0 \leq a \leq n$, we add arcs $(x_a^i, z_a^i)$ and $(z_a^i, y_a^i)$. See Figure 2 for an illustration.

- Furthermore, we attach terminals to the paths as follows: (shown using magenta edges in Figure 1 and Figure 2)
  - for every pair $(i, j)$ with $1 \leq i, j \leq \ell$, $i \neq j$, we add arcs $(s^0_{0-n}, x_{i0}^j)$ and $(y_{i0}^j, t^0_{0-n})$;
  - for every $1 \leq i \leq \ell$ we add arcs $(s^0_{0-n}, z_{i0}^i)$ and $(z_{i0}^i, t^0_{0-n})$;
  - for every pair $(i, j)$ with $1 \leq i < j \leq \ell$ we add arcs $(s^1_{n-0}, x_{i0}^j)$ and $(y_{i0}^j, t^1_{n-0})$;
  - for every pair $(i, j)$ with $\ell \geq i > j \geq 1$ we add arcs $(s^1_{n-0}, x_{i0}^j)$ and $(y_{i0}^j, t^1_{n-0})$.

- For every pair $(i, j)$ with $1 \leq i < j \leq \ell$ we introduce an acyclic $n \times n$ grid $P_{i,j}$ with vertices $p_{a,b}^{i,j}$ for $1 \leq a \leq n, b \leq n$ and arcs $(p_{a,b}^{i,j}, p_{a+1,b}^{i,j})$ for every $1 \leq a < n$ and $1 \leq b < n$, as well as $(p_{a,b}^{i,j}, p_{a,b+1}^{i,j})$ for every $1 \leq a \leq n$ and $1 \leq b < n$. We call this grid $P_{i,j}$ as the p-grid for the pair $(i, j)$. We set the vertex $p_{a,b}^{i,j}$ to be a light vertex if $v_{i,a}v_{j,b} \in E(G)$, and super-heavy otherwise. Finally, for every $1 \leq a \leq n$ we introduce the following arcs (shown as dotted in Figure 1):
  \[ (x_a^i, p_{a,1}^{i,j}), (p_{a,n}^{i,j}, y_{a-1}^{i,j}), (x_a^j, p_{1,a}^{i,j}), (p_{n,a}^{i,j}, y_{a-1}^{i,j}). \]
Figure 1 Illustration of the reduction for Directed Multicut With 4 Pairs. For $1 \leq i < j \leq \ell$, the grid $P_{i,j}$ is surrounded by the bidirectional paths $X_{i,j}$ on the left, $X_{j,i}$ on the top, $Y_{i,j}$ on the right and $Y_{j,i}$ on the bottom. Edges incident on terminals are shown in magenta. Green vertices are medium, orange vertices are heavy and white vertices are super-heavy. A desired solution is marked by red circles.
Figure 2 Illustration of the reduction for Directe... 4 Pairs. For every $1 \leq i < j \leq \ell$, the $z$-path $Z_i$, corresponding to the color class $i$ is surrounded by the bidirectional paths $X_{i,j}$ on the left and $Y_{i,j}$ on the right. Edges incident on terminals are shown in magenta. Green vertices are medium, orange vertices are heavy and white vertices are super-heavy.
8:12 FPT Inapproximability of Directed Cut and Connectivity Problems

This concludes the construction of the instance \((G', T')\) of \textsc{Directed Multicut With 4 Pairs}. Note that \(|V(G')| = (n + \ell)O(1)|\), and also \(G'\) can be constructed in \((n + \ell)O(1)|\) time.

2.2.2 Completeness of Lemma 11:
\textbf{Multicut of cost } \leq 29\ell^2
\textbf{Suppose that } \text{val}(\Gamma) = 1 \Rightarrow \text{val}(\Gamma) \geq \frac{1}{10}.
\text{Let } X \text{ be a solution to the instance } \Gamma \text{ of } \text{ Directed Multicut With 4 Pairs} \text{ such that weight of } X \text{ is } 29.5\ell^2 \text{. We now show that } \text{val}(\Gamma) \geq \frac{1}{10}.

\textbf{Observation 12.} Note that every super-heavy vertex has weight 100\ell^2 and hence } X \text{ cannot contain any super-heavy vertex.

\textbf{Lemma 13.} \([\star]\) For each } i \in \ell, \text{ the solution } X \text{ contains at least one heavy vertex from } Z_i.

\textbf{Lemma 14.} \([\star]\) For each } 1 \leq i < j \leq \ell, \text{ the solution } X \text{ contains at least one medium vertex from } X_{i,j} \text{ and at least one medium vertex from } Y_{i,j}.

\textbf{Definition 15.} An integer } i \in \ell \text{ is good if } X \text{ contains exactly one heavy vertex from the } z\text{-path for the color class } i, \text{ i.e., } |X \cap Z_i| = 1. \text{ In this case, we say that } v_{i,j}^1 \text{ be the unique vertex from the } z\text{-path for class } i \text{ in the solution } X.

\textbf{Lemma 16.} \([\star]\) Let \text{Good } = \{ \in \ell : \text{ i is good} \}. \text{ Then } |\text{Good}| \geq \frac{37\ell}{40}.

\textbf{Definition 17.} Let } 1 \leq i < j \leq \ell. \text{ We say that the pair } (i, j) \text{ is great if } X \text{ contains

\begin{itemize}
\item exactly one medium vertex from the } z\text{-path for the pair } (i, j)
\item exactly one medium vertex from the } y\text{-path for the pair } (i, j)
\item exactly one medium vertex from the } x\text{-path for the pair } (j, i)
\item exactly one medium vertex from the } y\text{-path for the pair } (j, i)
\item exactly one light vertex from the } p\text{-grid for the pair } (i, j)
\end{itemize}

\text{Let } \text{Good-Pairs } = \{ (i, j) : 1 \leq i < j \leq \ell, i, j \in \text{Good} \}

\textbf{Definition 18.} \([\star]\) Let } 1 \leq i < j \leq \ell. \text{ If both } i \text{ and } j \text{ are good, and the pair } (i, j) \text{ is great then } v_{i,j}^1 - v_{i,j}^1 \in E(G).

\textbf{Definition 19.} Let } 1 \leq i < j \leq \ell. \text{ We define } X_{i,j} = X \cap (X_{i,j} \cup X_{j,i} \cup Y_{i,j} \cup Y_{j,i} \cup P_{i,j})
Lemma 20. [⋆] Let $1 \leq i < j \leq \ell$ be such that $i, j \in \text{GOOD}$. Then either
- the pair $(i, j)$ is great and weight of $X_{i,j}$ is exactly $9B$, or
- weight of $X_{i,j}$ is at least $10B$

Lemma 21. [⋆] Let $\mathcal{E} = \{1 \leq i < j \leq \ell : i, j \in \text{GOOD} \text{ and } (i, j) \text{ is great}\}$. Then $|\mathcal{E}| \geq \frac{1}{10} \cdot \binom{\ell}{2}$

Consider the following $\ell$-vertex subgraph $C$: for each $i \in [\ell]$
- if $i \in [\ell]$ is good then add $v_{3i}$ to $C$,
- otherwise add any vertex from $V_i$ into $C$.

From Lemma 21 it follows that there are at least $\frac{1}{10} \cdot \binom{\ell}{2}$ edges in $G$ which have both endpoints in $C$, and hence $\text{val}(\Gamma) \geq \frac{1}{10}$

2.3 Finishing the proof of Theorem 2

We again prove by contrapositive. Suppose that, for some constant $\varepsilon > 0$ and for some computable function $f(p)$ independent of $n$, there exists an $f(p) \cdot n^{O(1)}$-time $(\frac{59}{58} - \varepsilon)$-approximation algorithm for Directed Multicut. Let us call this algorithm $A$.

We create an algorithm $B$ that can distinguish between the two cases of Corollary 9 with $b(\ell) = 1 - \frac{\log 10}{\log 13} = o(1)$. Our new algorithm $B$ works as follows. Given an instance $(G, H, V_1 \cup \cdots \cup V_\ell)$ of MCSI where $H = K_{\ell, \ell}$, the algorithm $B$ uses the reduction from Lemma 11 to create a Directed Multicut With 4 Pairs instance $(G', T')$ with 4 terminal pairs. $B$ then runs $A$ on this instance with $p = 29\ell^2$; if $A$ returns a solution $N$ of cost less than $29.5\ell^2$, then $B$ returns YES. Otherwise, $B$ returns NO.

To see that algorithm $B$ can indeed distinguish between the YES and NO cases, first observe that, in the YES case the completeness property of Lemma 11 guarantees that the optimal solution has cost at most $29\ell^2$. Since $A$ is a $(\frac{59}{58} - \varepsilon)$-approximation algorithm, it returns a solution of cost at most $(\frac{59}{58} - \varepsilon) \cdot 29\ell^2 < 29.5\ell^2$: this means that $B$ outputs YES. On the other hand, if $(G, H, V_1 \cup \cdots \cup V_\ell)$ is a NO instance, i.e., $\text{val}(\Gamma) < \frac{1}{10} = \ell^{b(\ell) - 1}$, then the soundness property of Lemma 22 guarantees that the optimal solution in $G'$ has cost more than $29.5\ell^2$ (which is greater than $(\frac{59}{58} - \varepsilon) \cdot 29\ell^2$) and hence $B$ correctly outputs NO.

Finally, observe that the running time of $B$ is $f(p) \cdot |V(G')|^{O(1)}$ plus the $(|V(G)| + \ell)^{O(1)}$ time needed to construct $G'$. Since $|V(G')| = (|V(G)| + \ell)^{O(1)}$ and $p = O(\ell^2)$ it follows that the total running time is $g(\ell) \cdot |V(G)|$ for some computable function $g$. Hence, from Corollary 9, Gap-ETH is violated.

3 FPT inapproximability for DSN_{Planar}

3.1 $(2 - \varepsilon)$-hardness for FPT approximation under Gap-ETH

The goal of this section is to show the following theorem:

Theorem 3. Under Gap-ETH, for any $\varepsilon > 0$ and any computable function $f$, there is no $f(k) \cdot n^{O(1)}$ time algorithm that computes a $(2 - \varepsilon)$-approximation for DSN_{PLANAR}.

3.1.1 Reduction from Colored Biclique to DSN_{Planar}

Lemma 22. For every constant $\gamma > 0$, there exists a polynomial time reduction that, given an instance $\Gamma = (G, H, V_1 \cup \cdots \cup V_\ell, W_1, W_2, \ldots, W_\ell)$ of MCSI where the supergraph $H$ is $K_{\ell, \ell}$, produces an instance $(G', D')$ of DSN_{PLANAR}, such that
(Completeness) If \( \text{val}(\Gamma) = 1 \), then there exists a planar network \( N \subseteq G' \) of cost \( 2(1 + \gamma^{1/5}) \) that satisfies all demands.

(Soundness) If \( \text{val}(\Gamma) < \gamma \), then every network \( N \subseteq G' \) that satisfies all demands has cost more than \( 2(2 - 4\gamma^{1/5}) \).

(Parameter Dependency) The number of demand pairs \( k = |D'| \) is \( 2^\ell \).

Lemma 22 is proven as follows: we construct the DSN\textsubscript{Planar} instance in Section 3.1.1.2. The proofs of completeness and soundness of the reduction are deferred to the full version [10]. First, we construct a “path gadget” which we use repeatedly in our construction.

3.1.1.1 Constructing a directed “path” gadget

For every integer \( n \) we define the following gadget \( P_n \) which contains \( 2n \) vertices (see Figure 3). Since we need many of these gadgets later on, we will denote vertices of \( P_n \) by \( P_n(v) \) etc., in order to be able to distinguish vertices of different gadgets. All edges will have the same weight \( B \), which we will fix later during the reductions. The gadget \( P_n \) is constructed as follows: \( P_n \) has a directed path of one edge corresponding to each \( i \in [n] \). This is given by \( P_n(0_i) \rightarrow P_n(1_i) \).

![Figure 3](image-url) The construction of the path gadget for \( P_n \). Note that the gadget has \( 2n \) vertices. Each edge of \( P_n \) has the same weight \( B \).

3.1.1.2 Construction of the DSN\textsubscript{Planar} instance

We give a reduction which transforms an instance \( G = (V, E) \) of MCSI\((K_\ell, \ell)\) into an instance of DSN which has \( 2\ell \) demand pairs and an optimum which is planar. Let the partition of \( V \) into color classes be given by \( \{V_1, V_2, \ldots, V_\ell, W_1, W_2, \ldots, W_\ell\} \). Without loss of generality (by adding isolated vertices if necessary), we can assume that each color class has the same number of vertices. Let \( |V_i| = |W_i| = n' \) for each \( 1 \leq i \leq \ell \). Then \( n = |V(G)| = 2n'\ell \). For each \( 1 \leq i, j \leq \ell \) we denote by \( E_{i,j} \) the set of edges with one end-point in \( V_i \) and other in \( W_j \).

We design two types of gadgets: the main gadget and the secondary gadget. The reduction from MCSI\((K_\ell, \ell)\) represents each edge set \( E_{i,j} \) with a main gadget \( M_{i,j} \). This is done as follows: each main gadget is a copy of the path gadget \( P_{|E_{i,j}|} \) from Section 3.1.1.1 with \( B = \frac{2}{7\ell} \), i.e., there is a row in \( M_{i,j} \) corresponding to each edge in \( E_{i,j} \). Each main gadget
is surrounded by four secondary gadgets: on the top, right, bottom and left. Each of these
gadgets are copies of the path gadget from Section 3.1.1.1 with $B = 0$:

- For each $1 \leq i \leq \ell + 1, 1 \leq j \leq \ell$ the horizontal gadget $HS_{i,j}$ is a copy of $\mathcal{P}_{|W_i|}$
- For each $1 \leq i \leq \ell, 1 \leq j \leq \ell + 1$ the vertical gadget $VS_{i,j}$ is a copy of $\mathcal{P}_{|V_i|}$

We refer to Figure 4 (bird’s-eye view) and Figure 5 (zoomed-in view) for an illustration
of the reduction. Fix some $1 \leq i, j \leq \ell$. The main gadget $M_{i,j}$ has four secondary
gadgets surrounding it:

- Above $M_{i,j}$ is the vertical secondary gadget $VS_{i,j+1}$
- On the right of $M_{i,j}$ is the horizontal secondary gadget $HS_{i+1,j}$
- Below $M_{i,j}$ is the vertical secondary gadget $VS_{i,j}$
- On the left of $M_{i,j}$ is the horizontal secondary gadget $HS_{i,j}$

Hence, there are $\ell(\ell + 1)$ horizontal secondary gadgets and $\ell(\ell + 1)$ vertical secondary gadgets.

**Red intra-gadget edges:** Fix $(i, j)$ such that $1 \leq i, j \leq \ell$. Recall that $M_{i,j}$ is a copy of $\mathcal{P}_{|E_{i,j}|}$
with $B = \frac{\ell - 1}{2}$ and each of the secondary gadgets are copies of $\mathcal{P}_{w}$ with $B = 0$. With slight abuse
of notation, we assume that the rows of $M_{i,j}$ are indexed by the set $\{(x, y) : (x, y) \in E_{i,j}, x \in W_i, y \in V_j\}$. We add the following edges (in red color) of weight 0:
for each $(x, y) \in E_{i,j}$
- Add the edge $VS_{i,j+1}(1_x) \rightarrow M_{i,j}(0_{(x,y)})$. These edges are called top-red edges incident
  on $M_{i,j}$.
- Add the edge $HS_{i,j+1}(1_y) \rightarrow M_{i,j}(0_{(x,y)})$. These edges are called left-red edges incident
  on $M_{i,j}$.
- Add the edge $M_{i,j}(1_{(x,y)}) \rightarrow HS_{i+1,j}(0_y)$. These edges are called right-red edges incident
  on $M_{i,j}$.
- Add the edge $M_{i,j}(1_{(x,y)}) \rightarrow VS_{i,j}(0_x)$. These edges are called bottom-red edges incident
  on $M_{i,j}$.

These are called the intra-gadget edges incident on $M_{i,j}$.

Introduction the following $4\ell$ vertices (which we call border vertices):

- $a_1, a_2, \ldots, a_{\ell}$
- $b_1, b_2, \ldots, b_{\ell}$
- $c_1, c_2, \ldots, c_{\ell}$
- $d_1, d_2, \ldots, d_{\ell}$

**Orange edges:** For each $i \in [\ell]$ add the following edges (shown as orange in Figure 4) with
weight $\frac{2^{3/2}}{4\ell}$:

- $a_i \rightarrow VS_{i,\ell+1}(0_v)$ for each $v \in V_i$. These are called top-orange edges.
- $VS_{i,1}(0_v) \rightarrow b_i$ for each $v \in V_i$. These are called bottom-orange edges.
- $c_j \rightarrow HS_{1,j}(0_w)$ for each $w \in W_j$. These are called left-orange edges.
- $HS_{\ell+1,j}(1_w) \rightarrow d_j$ for each $w \in W_j$. These are called right-orange edges.

Finally, the set of demand pairs $\mathcal{D}'$ is given by:

- **Type I:** the pairs $(a_i, b_i)$ for each $1 \leq i \leq \ell$.
- **Type II:** the pairs $(c_j, d_j)$ for each $1 \leq j \leq \ell$.

Clearly, the total number of demand pairs is $k = |\mathcal{D}'| = 2\ell$. Let the final graph constructed
be $G'$. Note that $G'$ has size $N = (n + \ell)^{O(1)}$ and can be constructed in $(n + \ell)^{O(1)}$ time.
It is also easy to see that $G'$ is actually a DAG.
Figure 4 A bird’s-eye view of the instance of $G'$ with $\ell = 3$ and $n' = 4$ (see Figure 5 for a zoomed-in view). Additionally we have some red edges between each main gadget and the four secondary gadgets surrounding it which are omitted in this figure for clarity (they are shown in Figure 5 which gives a more zoomed-in view).
Figure 5 A zoomed-in view of the main gadget $M_{i,j}$ surrounded by four secondary gadgets: vertical gadget $VS_{i,j+1}$ on the top, horizontal gadget $HS_{i,j}$ on the left, vertical gadget $VS_{i,j}$ on the bottom and horizontal gadget $HS_{i+1,j}$ on the right. Each of the secondary gadgets is a copy of the uniqueness gadget $U_n$ (see Section 3.1.1.1) and the main gadget $M_{i,j}$ is a copy of the uniqueness gadget $U_{|S_{i,j}|}$. The only inter-gadget edges are the red edges: they have one end-point in a main gadget and the other end-point in a secondary gadget. We have shown four such red edges which are introduced for every $(x,y) \in E_{i,j}$. 
3.1.2 Finishing the proof of Theorem 3

We can now easily prove Theorem 3 by combining Lemma 22 and Corollary 10.

Proof of Theorem 3. We again prove by contrapositive. Suppose that, for some constant \( \varepsilon > 0 \) and for some function \( f(k) \) independent of \( n \), there exists an \( f(k) \cdot N^O(1) \)-time \((2 - \varepsilon)\)-approximation algorithm for \( \text{DSN}_{\text{PLANAR}} \) where \( k \) is the number of terminal pairs and \( N \) is the size of the instance. Let us call this algorithm \( A \).

Given \( \varepsilon > 0 \), it is easy to see that there exists a sufficiently small \( \gamma^* = \gamma^*(\varepsilon) \) such that \( \frac{2(2 - 4\gamma^{*1/5})}{2(1 + 2\gamma^{*1/5})} \geq (2 - \varepsilon) \). We create an algorithm \( B \) that can distinguish between the two cases of Corollary 10 with \( h(\ell) = 1 - \frac{\log(1/\gamma^*)}{\log k} = o(1) \). Our new algorithm \( B \) works as follows. Given an instance \((G, H, V_1 \cup \cdots \cup V_k, W_1 \cup \cdots \cup W_\ell)\) of MCSI of size \( n \) where \( H = K_{\ell, \ell} \), the algorithm \( B \) uses the reduction from Lemma 22 to create in \((n + \ell)O(1)\) time a \( \text{DSN}_{\text{PLANAR}} \) instance on the graph \( G' \) with \( k = 2\ell \) terminal pairs and size \( N = (\ell + n)O(1) \). The algorithm \( B \) then runs \( A \) on this instance; if \( A \) returns a solution \( N \) of cost at most \( 2(2 - 4\gamma^{*1/5}) \), then \( B \) returns YES. Otherwise, \( B \) returns NO.

We now show that the algorithm \( B \) can indeed distinguish between the YES and NO cases of Corollary 10. In the YES case, i.e., \( \text{val}(\Gamma) = 1 \), the completeness property of Lemma 22 guarantees that the optimal planar solution has cost at most \( 2(1 + \gamma^{1/5}) \). Since \( A \) is a \((2 - \varepsilon)\)-approximation algorithm, it returns a solution of cost at most \( 2(1 + \gamma^{1/5}) \cdot (2 - \varepsilon) \leq 2(2 - 4\gamma^{*1/5}) \) where the inequality comes from our choice of \( \gamma^* \); this means that \( B \) outputs YES. On the other hand, in the NO case, i.e., \( \text{val}(\Gamma) < \gamma \), the soundness property of Lemma 22 guarantees that the optimal solution (and hence the planar optimal solution as well, if it exists) in \( G' \) has cost more than \( 2(2 - 4\gamma^{*1/5}) \), which implies that \( B \) outputs NO.

Finally, observe that the running time of \( B \) is \( f(k) \cdot N^O(1) + \text{poly}(\ell + n)O(1) \) which is bounded by \( f'(\ell) \cdot n^O(1) \) for some computable function \( f' \) since \( k = 2\ell \) and \( N = (n + \ell)O(1) \). Hence, from Corollary 10, Gap-ETH is violated.

3.2 Lower Bounds for FPT Approximation Schemes for \( \text{DSN}_{\text{PLANAR}} \)

We obtain the following result regarding the parameterized complexity of \( \text{DSN}_{\text{PLANAR}} \) parameterized by \( k + p \).

\begin{itemize}
  \item \textbf{Theorem 4.} \([x]\) The \( \text{DSN}_{\text{PLANAR}} \) problem is \( \text{W}[1]\)-hard parameterized by \( p + k \). Moreover, under ETH, for any computable function \( f \) and any \( \varepsilon > 0 \)
    \begin{itemize}
      \item There is no \( f(k, p) \cdot n^{o(k + \sqrt{p + 1/\varepsilon})} \) time algorithm for \( \text{DSN}_{\text{PLANAR}} \), and
      \item There is no \( f(k, \varepsilon, p) \cdot n^{o(k + \sqrt{p + 1/\varepsilon})} \) time algorithm which computes a \((1 + \varepsilon)\)-approximation for \( \text{DSN}_{\text{PLANAR}} \).
    \end{itemize}
\end{itemize}

4 Lower Bounds for FPT Approximation Schemes for \( \text{SCSS}_{\text{PLANAR}} \)

We obtain the following result regarding the parameterized complexity of \( \text{DSN}_{\text{PLANAR}} \) parameterized by \( k + p \).

\begin{itemize}
  \item \textbf{Theorem 5.} \([x]\) The \( \text{SCSS}_{\text{PLANAR}} \) problem is \( \text{W}[1]\)-hard parameterized by \( p + k \). Moreover, under ETH, for any computable function \( f \) and any \( \varepsilon > 0 \)
    \begin{itemize}
      \item there is no \( f(k, p) \cdot n^{o(\sqrt{k + p})} \) time algorithm for \( \text{SCSS}_{\text{PLANAR}} \), and
      \item there is no \( f(k, \varepsilon, p) \cdot n^{o(\sqrt{k + p + 1/\varepsilon})} \) time algorithm which computes an \((1 + \varepsilon)\)-approximation for \( \text{SCSS}_{\text{PLANAR}} \).
    \end{itemize}
\end{itemize}
References


8:20  FPT Inapproximability of Directed Cut and Connectivity Problems

TIGHT BOUNDS FOR PLANAR STRONGLY CONNECTED 
STEINER SUBGRAPH WITH FIXED NUMBER OF TERMINALS 
(AND EXTENSIONS)*
RAJESH H. CHITNIS1, ANDREAS E. FELDMANN2, MOHAMMADTAGHI HAJIAGHAYI3, 
AND DANIEL MARX4

Abstract. Given a vertex-weighted directed graph \( G = (V, E) \) and a set \( T = \{t_1, t_2, \ldots, t_k\} \) of \( k \) terminals, the objective of the STRONGLY CONNECTED STEINER SUBGRAPH (SCSS) problem is to find a vertex set \( H \subseteq V \) of minimum weight such that \( G[H] \) contains a \( t_i \rightarrow t_j \) path for each \( i \neq j \). The problem is NP-hard, but Feldman and Ruhl [SIAM J. Comput., 36 (2006), pp. 543–561] gave a novel \( n^{O(k)} \) algorithm for the SCSS problem, where \( n \) is the number of vertices in the graph and \( k \) is the number of terminals. We explore how much easier the problem becomes on planar directed graphs. Our main algorithmic result is a \( 2^{O(k)} \cdot n^{O(\sqrt{k})} \) algorithm for planar SCSS, which is an improvement of a factor of \( O(\sqrt{k}) \) in the exponent over the algorithm of Feldman and Ruhl. Our main hardness result is a matching lower bound for our algorithm: we show that planar SCSS does not have an \( f(k) \cdot n^{o(\sqrt{k})} \) algorithm for any computable function \( f \), unless the exponential time hypothesis (ETH) fails. To obtain our algorithm, we first show combinatorially that there is a minimal solution whose treewidth is \( O(\sqrt{k}) \), and then use the dynamic-programming based algorithm for finding bounded-treewidth solutions due to Feldmann and Marx [The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems, preprint, https://arxiv.org/abs/1707.06908]. To obtain the lower bound matching the algorithm, we need a delicate construction of gadgets arranged in a gridlike fashion to tightly control the number of terminals in the created instance. The following additional results put our upper and lower bounds in context: our \( 2^{O(k)} \cdot n^{O(\sqrt{k})} \) algorithm for planar directed graphs can be generalized to graphs excluding a fixed minor. Additionally, we can obtain this running time for the problem of finding an optimal planar solution even if the input graph is not planar. In general graphs, we cannot hope for such a dramatic improvement over the \( n^{O(k)} \) algorithm of Feldman and Ruhl: assuming ETH, SCSS in general graphs does not have an \( f(k) \cdot n^{o(k)} \) algorithm for any computable function \( f \). Feldman and Ruhl generalized their \( n^{O(k)} \) algorithm to the more general DIRECTED STEINER NETWORK (DSN) problem; here the task is to find a subgraph of minimum weight such that for every source \( s_i \) there is a path to the corresponding terminal \( t_i \). We show that, assuming ETH, there is no \( f(k) \cdot n^{o(k)} \) time algorithm for DSN on acyclic planar graphs. All our lower bounds hold for the integer weighted edge version, while the algorithm works for the more general unweighted vertex version.

Key words. strongly connected Steiner subgraph, FPT algorithms, directed Steiner network, exponential time hypothesis, planar graphs

AMS subject classifications. 05C85, 68W40

DOI. 10.1137/18M122371X

*Received by the editors October 29, 2018; accepted for publication (in revised form) November 25, 2019; published electronically March 25, 2020. A preliminary version of this paper appeared in SODA, SIAM, Philadelphia, 2014, pp. 1782–1801 [19].

1Department of Computer Science, University of Birmingham, Birmingham, B152TT, UK (rajeshchitnis@gmail.com).
2Department of Applied Math, Charles University in Prague, 118 00 Prague, Czech Republic (feldmann.a.e@gmail.com).
3Computer Science Department, University of Maryland, College Park, MD, 20742 (hajiagha@cs.umd.edu).
4Max Planck Institute for Informatics, Saarbrucken, Saarland, Germany (dmars@mpi-inf.mpg.de).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
1. Introduction. The Steiner Tree (ST) problem is one of the earliest and most fundamental problems in combinatorial optimization: given an undirected graph $G = (V, E)$ and a set $T \subseteq V$ of terminals, the objective is to find a tree of minimum size which connects all the terminals. The ST problem is believed to have been first formally defined by Gauss in a letter in 1836, and the first combinatorial formulation is attributed independently to Hakimi [43] and Levin [52] in 1971. The ST problem is known to be NP-complete, and was in fact part of Karp’s original list [49] of 21 NP-complete problems. In the directed version of the ST problem, called Directed Steiner Tree (DST), we are also given a root vertex $r$ and the objective is to find a minimum size arborescence which connects the root $r$ to each terminal from $T$. An easy reduction from Set Cover shows that the DST problem is also NP-complete.

Steiner-type problems arise in the design of networks. Since many networks are symmetric, the directed versions of Steiner-type problems were mostly of theoretical interest. However, in recent years, it has been observed [65, 67] that the connection cost in various networks such as satellite or radio networks are not symmetric. Therefore, directed graphs form the most suitable model for such networks. In addition, Ramanathan [65] also used the DST problem to find low-cost multicast trees, which have applications in point-to-multipoint communication in high bandwidth networks. We refer the interested reader to Winter [69] for a survey on applications of Steiner problems in networks.

In this paper we consider two well-studied Steiner-type problems in directed graphs, namely the Strongly Connected Steiner Subgraph (SCSS) and the Directed Steiner Network (DSN) problems. In the (vertex-unweighted) SCSS problem, given a directed graph $G = (V, E)$ and a set $T = \{t_1, t_2, \ldots, t_k\}$ of $k$ terminals, the objective is to find a set $S \subseteq V$ of minimum size such that $G[S]$ contains a $t_i \rightarrow t_j$ path for each $1 \leq i \neq j \leq k$. Thus, just as DST, the SCSS problem is another directed version of the ST problem, where all terminals need to be connected to each other. The (vertex-unweighted) DSN problem generalizes both DST and SCSS: given a directed graph $G = (V, E)$ and a set $T = \{(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\}$ of $k$ pairs of terminals, the objective is to find a set $S \subseteq V$ of minimum size such that $G[S]$ contains an $s_i \rightarrow t_i$ path for each $1 \leq i \leq k$. We first describe the known results for both SCSS and DSN before stating our results and techniques.

1.1. Previous work. Since both DSN and SCSS are NP-complete, one can try to design polynomial time approximation algorithms for these problems. An $\alpha$-approximation for DST implies a $2\alpha$-approximation for SCSS as follows: fix a terminal $t \in T$ and take the union of the solutions of the DST instances $(G, t, T \setminus t)$ and $(G_{rev}, t, T \setminus t)$, where $G_{rev}$ is the graph obtained from $G$ by reversing the orientations of all edges. The best known approximation ratio in polynomial time for SCSS is $k^\epsilon$ for any $\epsilon > 0$ [14]. A result of Halperin and Krauthgamer [44] implies SCSS has no $\Omega(\log^{2-\epsilon} n)$-approximation for any $\epsilon > 0$, unless NP has quasi-polynomial Las Vegas algorithms. For the more general DSN problem, the best approximation ratio known is $n^{2/3+\epsilon}$ for any $\epsilon > 0$. Berman et al. [4] showed that DSN has no $\Omega(2^{\log^{1-\epsilon} n})$-approximation for any $0 < \epsilon < 1$, unless NP has quasi-polynomial-time algorithms.

Rather than finding approximate solutions in polynomial time, one can look for exact solutions in time that are still better than the running time obtained by brute force algorithms. For (unweighted versions of) both the SCSS and DSN problems, brute force can be used to check in time $n^{O(p)}$ if a solution of size at most $p$ exists: one can go through all sets of size at most $p$. A more efficient algorithm would have runtime $f(p) \cdot n^{O(1)}$, where $f$ is some computable function depending only on $p$. A
problem is said to be fixed-parameter tractable (FPT) with a particular parameter $p$ if it admits such an algorithm; see [23, 28, 37, 62] for more background on FPT algorithms. A natural parameter for our considered problems is the number $k$ of terminals or terminal pairs; with this parameterization, it is not even clear if there is a polynomial time algorithm for every fixed $k$, much less if the problem is FPT. It is known that ST on undirected graphs is FPT parameterized by the number $k$ of terminals: the classical algorithm of Dreyfus and Wagner [29] solves the problem in time $3^k \cdot n^{O(1)}$. The running time was recently improved to $2^k \cdot n^{O(1)}$ by Björklund et al. [5]. The same algorithms work for DST as well.

For the SCSS and DSN problems, we cannot expect fixed-parameter tractability: Guo, Niedermeier, and Suchý [42] showed that SCSS is W[1]-hard parameterized by the number of terminals $k$, and DSN is W[1]-hard parameterized by the number of terminal pairs $k$. In fact, it is not even clear how to solve these problems in polynomial time for small fixed values of the number $k$ of terminals/pairs. The case of $k = 1$ in DSN is the well-known shortest path problem in directed graphs, which is known to be polynomial time solvable. For the case $k = 2$ in DSN, an $O(n^3)$ algorithm was given by Li, McCormick, and Simchi-Levi [53] which was later improved to $O(mn + n^2 \log n)$ by Natsu and Fang [61]. The question regarding the existence of a polynomial time algorithm for DSN when $k = 3$ was open. Feldman and Ruhl [35] solved this question by giving an $n^{O(k)}$ algorithm for DSN, where $k$ is the number of terminal pairs. They first designed an $n^{O(k)}$ algorithm for SCSS, where $k$ is the number of terminals, and used it as a subroutine in the algorithm for the more general DSN problem.

1.2. Our results and techniques. Given the amount of attention the planar version of Steiner-type problems received from the viewpoint of approximation (see, e.g., [2, 3, 11, 27, 32]) and the availability of techniques for parameterized algorithms on planar graphs (see, e.g., [6, 25, 40, 50, 60]), it is natural to explore SCSS and DSN restricted to planar graphs. In general, one can have the expectation that the problems restricted to planar graphs become easier, but sophisticated techniques might be needed to exploit planarity. In particular, a certain square root phenomenon was observed for a wide range of algorithmic problems: the exponent of the running time can be improved from $O(k)$ to $O(\sqrt{k})$ (or to $O(\sqrt{k} \log k)$) and lower bounds indicate that this improvement is essentially best possible [58, 50, 60, 51, 39, 24, 63, 59, 55, 1, 38]. Our main algorithmic result is also an improvement of this form.

Theorem 1.1. An instance $(G, T)$ of the vertex-weighted Strongly Connected Steiner Subgraph problem with $|G| = n$ and $|T| = k$ can be solved in $2^{O(k)} \cdot n^{O(\sqrt{k})}$ time, when the underlying undirected graph of $G$ is planar.

This algorithm presents a major improvement over the Feldman–Ruhl algorithm for SCSS in general graphs which runs in $n^{O(k)}$ time. A preliminary version of this paper [19] by a subset of the authors contained a complicated algorithm with a worse running time of $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$. It relied on modifying the Feldman–Ruhl token game, and then using the excluded grid theorem for planar graphs followed by treewidth-based techniques. We briefly give some of the intuition behind this algorithm and the original $n^{O(k)}$ algorithm of Feldman–Ruhl. The algorithm of Feldman–Ruhl for SCSS is based on defining a game with $2k$ tokens and costs associated with the moves of the tokens such that the minimum cost of the game is equivalent to the minimum cost of a solution of the SCSS problem; then the minimum cost of the game

---

1Planarity for directed graph problems refers to the underlying undirected graph being planar.
can be computed by exploring a state space of size $n^{O(k)}$. The $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ algorithm was obtained by generalizing the Feldman–Ruhl token game via introducing supermoves, which are sequences of certain types of moves. The generalized game still has a state space of $n^{O(k)}$, but it has the advantage that we can now give a bound of $O(k)$ on the number of supermoves required for the game (such a bound is not possible for the original version of the game). This gives an $O(k)$-sized summary of the token game, and hence has treewidth $O(\sqrt{k})$. However, this summary is “unlabeled,” i.e., we do not explicitly know which vertices occur where in the summary. Guessing by brute force requires $n^{O(k)}$ time, and the improvement to $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ is obtained by using an embedding theorem of Klein and Marx [50].

Unlike the $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ algorithm of [19], the $2^{O(k)} \cdot n^{O(\sqrt{k})}$ algorithm from Theorem 1.1 does not depend on the Feldman–Ruhl algorithm. It is conceptually much simpler: first we show combinatorially (see Lemma 2.2) that there is a minimal solution whose treewidth is $O(\sqrt{k})$, and then use the dynamic-programming-based algorithm for finding bounded-treewidth solutions for DSN due to Feldmann and Marx [36, Theorem 5]. The simplicity of our new approach also allows transparent generalizations in two directions:

- **From planar to $H$-minor-free graphs**: we may use the excluded grid minor theorem for $H$-minor-free graphs [26] instead of the excluded grid minor theorem for planar graphs [66] to prove the existence of a minimal solution of treewidth $O(\sqrt{k})$, which again implies a $2^{O(k)} \cdot n^{O(\sqrt{k})}$ time algorithm.
- **Between restricted inputs and restricted solutions**: our algorithm only uses the $H$-minor-freeness of an optimum solution, and not of the whole input graph. Thus, only the existence of one optimum $H$-minor-free solution in an otherwise unrestricted input graph is enough to show that some optimum solution (which might not necessarily be $H$-minor-free) can be found in $2^{O(k)} \cdot n^{O(\sqrt{k})}$ time.\footnote{This can be thought of as an intermediate requirement between the two extremes of either forcing the input graph itself to be $H$-minor-free versus the other extreme of finding an optimum solution which is $H$-minor-free. There has been some recent work in this direction [17, 18].}

Can we get a better speedup in planar graphs than the improvement from $O(k)$ to $O(\sqrt{k})$ in the exponent of $n$? Our main hardness result matches our algorithm: it shows that $O(\sqrt{k})$ is best possible under the exponential time hypothesis (ETH).

**Theorem 1.2.** The edge-unweighted version of the SCSS problem is W[1]-hard parameterized by the number of terminals $k$, even when the underlying undirected graph is planar. Moreover, under ETH, the SCSS problem on planar graphs cannot be solved in $f(k) \cdot n^{o(\sqrt{k})}$ time, where $f$ is any computable function, $k$ is the number of terminals, and $n$ is the number of vertices in the instance.

This also answers the question of Guo, Niedermeier, and Suchý [42], who showed the W[1]-hardness of these problems on general graphs and left the fixed-parameter tractability status on planar graphs as an open question. Recall that ETH has the consequence that $n$-variable 3SAT cannot be solved in time $2^{o(n)}$ [45, 46]. There are relatively few parameterized problems that are W[1]-hard on planar graphs [7, 13, 33, 58]. The reason for the scarcity of such hardness results is mainly because for most problems, the fixed-parameter tractability of finding a solution of size $k$ in a planar graph can be reduced to a bounded-treewidth problem by standard layering techniques. However, in our case the parameter $k$ is the number of terminals, hence such a simple reduction to the bounded-treewidth case does not seem to be possible.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Our reduction is from the Grid Tiling problem formulated by Marx [56, 58] (see also [23]), which is a convenient starting point for parameterized reductions for planar problems. For our reduction we need to construct two types of gadgets, namely, the connector gadget and main gadget, which are then arranged in a gridlike structure (see Figure 2). The main technical part of the reduction is the structural result regarding the existence and construction of particular types of connector gadgets and main gadgets (Lemmas 3.3 and 3.6). Interestingly, the construction of the connector gadget poses a greater challenge: here we exploit in a fairly delicate way the fact that the \( t_i \sim t_j \) and the reverse \( t_j \sim t_i \) paths appearing in the solution subgraph might need to share edges to reduce the weight.

We present additional results that put our algorithm and lower bound for SCSS in a wider context. Given our speedup for SCSS in planar graphs, one may ask if it is possible to get any similar speedup in general graphs. Our next result shows that the \( n^{O(k)} \) algorithm of Feldman–Ruhl is almost optimal in general graphs.

**Theorem 1.3.** **Under ETH, the edge-unweighted version of the SCSS problem cannot be solved in time \( f(k) \cdot n^{o(k/\log k)} \), where \( f \) is an arbitrary computable function, \( k \) is the number of terminals, and \( n \) is the number of vertices in the instance.**

Our proof of Theorem 1.3 is similar to the \( W[1] \)-hardness proof of Guo, Niedermeier, and Suchý [42]. They showed the \( W[1] \)-hardness of SCSS on general graphs parameterized by the number \( k \) of terminals by giving a reduction from \( k \)-Clique. However, this reduction uses "edge selection gadgets" and since a \( k \)-clique has \( \Theta(k^2) \) edges, the parameter is increased at least to \( \Theta(k^2) \). Combining with the result of Chen et al. [15] regarding the nonexistence of an \( f(k) \cdot n^{o(k)} \) algorithm for \( k \)-Clique under ETH, this gives a lower bound of \( f(k) \cdot n^{o(\sqrt{k})} \) for SCSS on general graphs. To avoid the quadratic blowup in the parameter and thereby get a stronger lower bound, we use the Partitioned Subgraph Isomorphism (PSI) problem as the source problem of our reduction. For this problem, Marx [57] gave an \( f(k) \cdot n^{o(k/\log k)} \) lower bound under ETH, where \( k = |E(G)| \) is the number of edges of the subgraph \( G \) to be found in graph \( H \). The reduction of Guo, Niedermeier, and O. Suchý [42] from Clique can be turned into a reduction from PSI which uses only \( |E(G)| \) edge selection gadgets, and hence the parameter is \( \Theta(|E(G)|) \). Then the lower bound of \( f(k) \cdot n^{o(k/\log k)} \) transfers from PSI to SCSS. A natural question is whether we can close the \( O(\log k) \) factor in the exponent; however, our reduction is from the PSI problem and the best known lower bound for PSI also has such a gap [57]. Note that there are many other parameterized problems for which the only known way of proving almost tight lower bounds is by a similar reduction from PSI, and hence an \( O(\log k) \) gap appears for these problems as well [59, 47, 20, 48, 22, 9, 41, 10, 12, 34, 54, 8, 21, 18, 31, 64].

Even though Feldman and Ruhl were able to generalize their \( n^{O(k)} \) time algorithm from SCSS to DSN, we show that, surprisingly, such a generalization is not possible for our \( 2^{O(k)} \cdot n^{O(\sqrt{k})} \) time algorithm for planar SCSS.

**Theorem 1.4.** **The edge-unweighted version of the DSN problem is \( W[1] \)-hard parameterized by the number \( k \) of terminal pairs, even when the input is restricted to planar directed acyclic graphs (planar DAGs). Moreover, there is no \( f(k) \cdot n^{o(k)} \) time algorithm for any computable function \( f \), unless the ETH fails.**

This implies that the Feldman–Ruhl algorithm for DSN is optimal, even on planar DAGs. As in our lower bound for planar SCSS, the proof is by a reduction from an instance of the \( k \times k \) Grid Tiling problem. However, unlike in the reduction to SCSS where we needed \( O(k^4) \) terminals, the reduction to DSN needs only \( O(k) \) pairs.
of terminals (see Figure 6). Since the parameter blowup is linear, the $f(k) \cdot n^{o(k)}$ lower bound for GRID TILING from [56] transfers to DSN.

**Remark 1.5.** All our hardness results (Theorems 1.2, 1.3, and 1.4) are presented for weighted-edge versions with polynomially bounded integer weights (including edges with weight zero). By splitting each edge of weight $W$ into $W$ edges of weight one, all the results also hold for the unweighted-edge version. Our algorithm (Theorem 1.1) is presented for the weighted-vertex version. Appendix A shows that the unweighted-vertex version is more general than the weighted-edge version. Hence all our lower bounds also hold for the (un)weighted-vertex version too.

Finally, instead of parameterizing by the number of terminals, we can consider parameterization by the number of edges/vertices of the solution. Let us briefly and informally discuss this parameterization. Note that the number of terminals is a lower bound on the number of edges/vertices of the solution (up to a factor of 2 in the case of DSN parameterized by the number of edges), thus fixed-parameter tractability could be easier to obtain by parameterizing with the number of edges/vertices. However, our lower bound for SCSS on general graphs (as well as the W[1]-hardness of Guo, Niedermeier, and Suchý [42]) actually proves hardness also with these parameterizations, making fixed-parameter tractability unlikely. On the other hand, it follows from standard techniques that both SCSS and DSN are FPT on planar graphs when parameterizing by the number $k$ of edges/vertices in the solution. The main argument here is that the solution is fully contained in the $k$-neighborhood of the terminals, whose number is at most $2k$. It is known that the $k$-neighborhood of $O(k)$ vertices in a planar graph has treewidth $O(k)$, and thus one can use standard techniques on bounded-treewidth graphs (dynamic programming or Courcelle’s theorem). Alternatively, at least in the unweighted case, one can formulate the problem as a first order formula of size depending only on $k$ and then invoke the result of Frick and Grohe [40] stating that such problems are FPT. Therefore, as fixed-parameter tractability is easy to establish on planar graphs, the challenge here is to obtain optimal dependence on $k$. One would expect that subexponential dependence on $k$ (e.g., $2^{O(\sqrt{k})}$ or $k^{O(\sqrt{k})}$) should be possible at least for SCSS, but this is not yet fully understood even for undirected ST [63]. A slightly different parameterization is to consider the number $k$ of non-terminal vertices in the solution, which can be much smaller than the number of terminals. This leads to problems of a somewhat different flavor; see, e.g., [30, 48].

**1.3. Further related work.** Subsequent to the conference version [19] of this paper, there have been several related results. Chitnis et al. [16] considered a variant of SCSS with only 2 terminals but with a requirement of multiple paths. Formally, in the 2-SCSS-$(k_1, k_2)$ problem we are given two vertices $s, t$ and the goal is to find a min weight subset $H \subseteq E(G)$ such that $H$ has $k_1, k_2$ paths from $s \leadsto t, t \leadsto s$, respectively. The objective function is given by $\text{cost}(H) = \sum_{e \in H} \phi(e) \cdot \text{cost}(e)$, where $\phi(e)$ is the maximum number of times $e$ appears on $s \leadsto t$ paths and $t \leadsto s$ paths. Chitnis et al. [16] showed that the 2-SCSS-$(k, 1)$ problem can be solved in $n^{O(k)}$ time for any $k \geq 1$, and has a $f(k) \cdot n^{o(k)}$ lower bound under ETH.

Suchy [68] introduced a generalization of DST and SCSS called the $q$-ROOT STEINER TREE ($q$-RST) problem. In this problem, given a set of $q$ roots and a set of $k$ leaves, the task is to find a minimum-cost network where the roots are in the same strongly connected component and every leaf can be reached from every root. Generalizing the token game of Feldman and Ruhl [35], Suchý [68] designed a $2^{O(q)} \cdot n^{O(k)}$ algorithm for $q$-RST.
Recently, Chitnis, Feldmann, and Manurangsi [18] considered the SCSS and DSN problems on bidirected graphs: these are directed graphs with the guarantee that for every edge \((u,v)\) the reverse edge \((v,u)\) exists and has the same weight. They showed that on bidirected graphs, the DSN problem stays \(W[1]\)-hard parameterized by \(k\) but SCSS becomes FPT (while still being NP-hard). In fact, under ETH, no \(f(k)\cdot n^{o(k/\log k)}\) time algorithm for DSN on bidirected graphs exists, and thus the problem is essentially as hard as for general directed graphs. For bidirected planar graphs, however, Chitnis, Feldmann, and Manurangsi [18] show that DSN can be solved in \(2^{O(k^{3/2} \log k)} n^{O(\sqrt{k})}\), which is in contrast to Theorem 1.4. Some FPT approximability and inapproximability results for SCSS and DSN were also shown in [18, 17].

**Pattern graphs and DSN:** The set of pairs \(\{(s_i,t_i) : i \in [k]\}\) in the input of DSN can be interpreted as a directed (unweighted) pattern graph on a set \(R = \bigcup_{i=1}^{k} \{s_i,t_i\}\) of terminals. For a graph class \(\mathcal{H}\), the \(\mathcal{H}\)-DSN problem takes as input a directed graph \(H \in \mathcal{H}\) on vertex set \(R\) and the goal is to find a minimum cost subgraph \(N \subseteq E(G)\) such that \(N\) has an \(s \rightarrow t\) path for each \((s,t) \in E(H)\). Thus for a fixed class \(\mathcal{H}\) of pattern graphs, the \(\mathcal{H}\)-DSN problem is a restricted special case of the general DSN problem, and it is possible that \(\mathcal{H}\)-DSN is FPT (for example, if \(\mathcal{H}\) is the class of out-stars). Feldmann and Marx [36] gave a complete dichotomy for which graph classes the \(\mathcal{H}\)-DSN problem is FPT or \(W[1]\)-hard parameterized by \(|R|\).

Given an instance of DSN with the pattern graph \(H = (R,A)\) on the terminal set \(R\) with \(|A| = k\), the algorithm of Feldman and Ruhl [35] runs in \(n^{O(k)}\) time. The \(f(k) \cdot n^{o(k)}\) lower bound under ETH for DSN in this paper (Theorem 1.4) has \(|A| = O(|R|)\). Hence, for the parameter \(|R|\) we have a lower bound of \(f(|R|) \cdot n^{o(|R|)}\) and an upper bound of \(n^{O(|R|^2)}\) (since \(|A| = O(|R|^2)\) in the worst case). Recently, Eiben et al. [31] essentially closed this gap by showing a lower bound of \(f(|R|) \cdot n^{o(|R|^2/\log |R|)}\) under ETH for DSN. They also gave an algorithm for DSN on bounded genus graphs: for graphs of genus \(g\), the algorithm runs in \(f(|R|) \cdot n^{O_g(|R|)}\) time, where \(O_g(\cdot)\) hides constants depending only on \(g\).

**2. Improved algorithm for SCSS on planar graphs.** In this section we describe the proof to Theorem 1.1, i.e., we present an algorithm to solve SCSS on planar graphs in \(2^{O(k)} \cdot n^{O(\sqrt{k})}\) time. The definitions of some of the graph-theoretic notions used in this section such as treewidth and minors are deferred to Appendix B to maintain the flow of presentation. The key is to analyze the structure of edge-minimal solutions, i.e., subgraphs of the input graph \(G\) (induced by some set \(S \subseteq V\)) containing all terminals for which no edge can be removed without also removing all \(s \rightarrow t\) paths for some terminal pair \((s,t)\). We show that for an edge-minimal solution \(M\) of the SCSS problem there is a vertex set \(W \subseteq V(M)\) of size \(O(k)\) such that, after removing \(W\) from \(M\), each component has constant treewidth. More formally, we define a \(W_M\)-component as a connected component of the (underlying undirected) graph induced by \(V(M) \setminus W\) in \(M\), and prove the following.

**LEMMA 2.1.** For any edge-minimal solution \(M\) to the edge-weighted SCSS problem there is a set of at most \(9k\) vertices \(W \subseteq V(M)\) for which every \(W_M\)-component has treewidth at most 2.

We defer the proof of Lemma 2.1 to section 2.1. First, we see how we can use Lemma 2.1 to bound the treewidth of the minimal solution \(M\).

**LEMMA 2.2.** If an edge-minimal solution \(M\) to edge-weighted SCSS is planar (or excludes some fixed minor), then its treewidth is \(O(\sqrt{k})\).
Proof. By the planar grid theorem \cite{66}, there is a constant $c_{\text{planar}}$ such that any planar graph $G$ with treewidth $c_{\text{planar}} \cdot \omega$ has a $\omega \times \omega$ grid minor. If the treewidth of $M$ is at least $c_{\text{planar}} \cdot 20\sqrt{k}$, then it follows that $M$ has a \([20\sqrt{k}] \times [20\sqrt{k}]\) grid minor $M'$. It is easy to see that $M'$ contains at least $\left\lfloor \frac{20\sqrt{k}}{3} \right\rfloor \cdot \left\lfloor \frac{20\sqrt{k}}{3} \right\rfloor$ (pairwise vertex-disjoint) grids of size $3 \times 3$. For each $k \geq 1$, one can easily verify that $\left\lfloor \frac{20\sqrt{k}}{3} \right\rfloor \geq 4\sqrt{k}$ and, hence, the number of pairwise vertex-disjoint $3 \times 3$ grids is at least $4\sqrt{k} \cdot 4\sqrt{k} \geq 16k$. By Lemma 2.1, there is a set of vertices $W$ of size $9k$ whose deletion makes every $W_M$-component have treewidth at most 2. Since $16k > 9k$, it follows that $W$ does not contain a vertex from at least one of the (pairwise vertex-disjoint) $16k$ grid minors of size $3 \times 3$ in $M$. Hence, there is a $W_M$-component, which contains a $3 \times 3$ grid minor and, hence, has treewidth at least 3, which is a contradiction.

For the case when the input graph is $H$-minor-free for some fixed graph $H$, we can instead use the excluded grid-minor theorem of Demaine and Hajiaghayi \cite{26}: for any fixed graph $H$, there is a constant $c_H$ (which depends only on $|H|$) such that any $H$-minor-free graph of treewidth at least $c_H \cdot \omega$ has an $\omega \times \omega$ grid as a minor. \hfill \Box

To prove Theorem 1.1, which is restated below, we invoke an algorithm of \cite{36} to find the optimum solution of bounded treewidth. The algorithm of \cite{36} is designed for the edge-weighted version, and we state below the corresponding statement for the more general unweighted vertex version (so that it may also be of future use).

**Theorem 2.3 (generalization of \cite{36, Theorem 5}).** If there is an optimum solution to an instance on $k$ terminals of the vertex-weighted version of SCSS which has treewidth at most $\omega$, then an optimum solution\(^3\) can be found in $2^{O(k \log \omega)} \cdot n^{O(\omega)}$ time.

Proof. In the given graph $G$, we start by subdividing each edge by adding a nonterminal vertex of weight 0 (note that this does not increase the treewidth). Let us call these vertices we have added dummy vertices, and the graph obtained at this point be $G'$. Note that each dummy vertex has in-degree one and out-degree one. Now we reduce the vertex-weighted version of SCSS to the edge-weighted version, using a standard reduction: substitute each nonterminal vertex $u \in G$ of weight $W$ with two new nonterminal vertices $u^-$ and $u^+$ and an edge $(u^-, u^+)$ of the same weight $W$. Every edge that had $u$ as its head will now have $u^-$ as its head instead, and every edge that had $u$ as its tail will now have $u^+$ as its tail. We set the weight of all these edges to be zero. Let the graph obtained after these modifications be $G'$.

Consider an optimum solution $S$ for the vertex-weighted version of SCSS, and without loss of generality we can assume that $S$ is minimal under vertex deletions (if it is not, then make it minimal by deleting unnecessary vertices). Let $S' = (S \cap T) \cup \{u^- \cdot u^+ : u \in S \cap (V \setminus T)\}$. We now show that the induced graph $G'[S']$ is an edge-minimal solution (with the same weight as that of $S$) for the edge-weighted version of SCSS: we do this by showing that deletion of any edge from $G'[S']$ creates a nonterminal source or a nonterminal sink which contradicts the fact that $S$ was a vertex-minimal solution for vertex-weighted version of SCSS. The construction of the graph $G'$ from $G$ implies that any edge $e$ in $G$ must be of either of the following two types:

- Without loss of generality,\(^4\) the edge is $(y, v^-)$ for some dummy vertex $y$ and some nonterminal $v \in G$, in which case deleting this edge makes the nonterminal $y$ to be a sink.

\(^3\)Not necessarily the same optimum solution as the one mentioned in the first part of this theorem. In particular, the actual optimum found by this algorithm might have treewidth much larger than $\omega$.

\(^4\)The other case is the edge being $(v^+, y)$ for some dummy vertex $y$ and some nonterminal $v \in G$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
• The edge is \((z^-, z^+)\) for some nonterminal \(z \in G\) in which case deleting this edge makes the nonterminal \(z^+\) a source and the nonterminal \(z^-\) a sink.

Note that \(G'\) is not necessarily planar (or \(H\)-minor-free) even if \(G\) is. However, the treewidth of \(G'[S']\) is at most twice the treewidth of \(G[S]\) since we can simply replace each nonterminal vertex \(u\) in the bags of the tree decomposition of \(N\) by the two vertices \(u^-\) and \(u^+\). Feldmann and Marx [36, Theorem 5] showed that if the optimum solution to an instance on \(k\) terminals of the edge-weighted version of SCSS has treewidth \(\omega\), then it can be found in \(2^{O(k^{1.5\log \omega})} \cdot n^{O(\omega)}\) time. Hence, the claimed running time for the vertex-weighted version follows.

Finally, we are now ready to prove Theorem 1.1, which is restated below.

**Theorem 1.1.** An instance \((G, T)\) of the vertex-weighted Strongly Connected Steiner Subgraph problem with \(|G| = n\) and \(|T| = k\) can be solved in \(2^{O(k)} \cdot n^{O(\sqrt{T})}\) time, when the underlying undirected graph of \(G\) is planar.

**Proof.** Consider a subgraph \(M\) of \(G\) induced by the optimum solution \(S \subseteq V\), which is also minimal, i.e., no edge of \(M\) can be removed without destroying the connectivity between some terminal pair \((s, t)\). By Lemma 2.2 we know that the treewidth of \(M\) is \(O(\sqrt{T})\). Hence, the claimed running time follows from Theorem 2.3.

Note that Lemma 2.2 only used the planarity (or \(H\)-minor-freeness) of \(M\), and not of the input graph. Hence, the algorithm of Theorem 1.1 also works for the weaker restriction of finding an optimal planar (or \(H\)-minor-free) solution in an otherwise unrestricted input graph, rather than finding an optimal solution in a planar (or \(H\)-minor-free, respectively) graph. It only remains to prove Lemma 2.1, which is done in the next section.

**2.1. Proof of Lemma 2.1.** Fix an arbitrary terminal \(r \in T\). It is easy to see (observed for example by Feldman and Ruhl [35]) that any minimal SCSS solution \(M\) is the union of an in-arborescence \(A_{in}\) and an out-arborescence \(A_{out}\), both having the same root \(r \in T\) and only terminals as leaves, since every terminal of \(T\) can be reached from \(r\), and conversely every terminal can reach \(r\) in \(M\). We construct the set \(W_M\) by including three different kinds of vertices. First, \(W_M\) contains every branching point of \(A_{in}\) and \(A_{out}\), i.e., every vertex with in-degree at least 2 in \(A_{in}\) and every vertex with out-degree at least 2 in \(A_{out}\). Since \(A_{in}\) and \(A_{out}\) are arborescences with at most \(k\) leaves (the terminals), they each have at most \(k\) branching points. Second, \(W_M\) contains all terminals of \(T\), which adds another \(k\) vertices to the set \(W_M\). The third kind of vertices in \(W_M\) is the following. Note that every component of the intersection of \(A_{in}\) and \(A_{out}\) forms a path (possibly consisting only of a single vertex), since every vertex of \(A_{in}\) has out-degree at most 1, while every vertex of \(A_{out}\) has in-degree at most 1. We call such a component a shared path. If a shared path contains a branching point or a terminal, we add the endpoints of the shared path to \(W_M\). For a branching point or terminal \(v\) on such a shared path, we can map the endpoints of the shared path to \(v\). This maps at most two endpoints of shared paths to each branching point or terminal, so that the number of vertices of the third kind in \(W_M\) is at most \(6k\) (as there are \(k\) terminals and at most \(2k\) branching points). Thus the total size of \(W_M\) is at most \(9k\).

We claim that every \(W_M\)-component consists of at most two interacting paths, one from \(A_{in}\) and one from \(A_{out}\). More formally, consider a \(u \rightarrow v\) path \(P\) of \(A_{in}\) such that \(u\) and \(v\) are either terminals or branching points of \(A_{in}\), and such that no internal vertex of \(P\) is a terminal or branching point of \(A_{in}\). We call any such path \(P\) an essential path of \(A_{in}\). Note that we ignore the branching points of \(A_{out}\) in this...
definition, and that the edge set of the arborescence $A_{in}$ is the disjoint union of the edge sets of its essential paths. Analogously we define the essential paths of $A_{out}$ as those $u \rightarrow v$ paths in $A_{out}$ for which $u$ and $v$ are terminals or branching points of $A_{out}$, and no internal vertices of $P$ are of such a type.

**Claim 2.4.** Every $W_M$-component contains edges of at most two essential paths, one from $A_{in}$ and one from $A_{out}$.

**Proof.** Any vertex at which two essential paths of the same arborescence intersect is a terminal or branching point. These vertices are in $W_M$ and therefore not contained in any $W_M$-component. Thus if a $W_M$-component $H$ contains at least two essential paths, then they either coincide on every edge of $H$, in which case the claim is clearly true, or $H$ contains the endpoint $v$ of a shared path, i.e., there are two essential paths, one from each arborescence, that both contain vertex $v$. We will show that there is only one pair of essential paths that can meet at an endpoint of a shared path in $H$, from which the claim follows.

In order to prove this, we label every essential $u \rightarrow v$ path $P$ of $A_{in}$ with those terminals $T_P \subseteq T$ that can reach the start vertex of $P$ in the in-arborescence, i.e., $t \in T_P$ if and only if there exists a $t \rightarrow u$ path in $A_{in}$. Note that no two essential paths of $A_{in}$ can have the same label. We also label any essential $u \rightarrow v$ path $Q$ of $A_{out}$ analogously, by setting the label $T_Q \subseteq T$ to be the terminals which can be reached from the end vertex of $Q$ in the out-arborescence, i.e., there is a $v \rightarrow t$ path in $A_{out}$ if and only if $t \in T_Q$. Even though no two essential paths of an individual arborescence have the same label, there can be pairs of essential paths from $A_{in}$ and $A_{out}$ with the same label. Let $P$ and $Q$ be essential paths of $A_{in}$ and $A_{out}$, respectively. We prove that if $P$ and $Q$ meet at an endpoint $v$ of a shared path, then $v \in W_M$ or $T_P = T_Q$.

Assume this is not the case so that $v \notin W_M$ and $T_P \neq T_Q$. Let $I$ be the shared path in the intersection of $A_{in}$ and $A_{out}$ for which $v$ is an endpoint. If $u$ is the other endpoint of $I$, assume without loss of generality that $I$ is a $u \rightarrow v$ path (the other case is symmetric). If there were any branching points or terminals on $I$ then $v \in W_M$, since $v$ would then be one of the third kind of vertices in $W_M$. As this is not the case, $I$ lies in the intersection of $P$ and $Q$, there are edges $e_v \in E(P)$ and $f_v \in E(Q)$ leaving $v$ such that $e_v \notin E(A_{out})$ and $f_v \notin E(A_{in})$, and there are edges $e_v \in E(P)$ and $f_v \in E(Q)$ entering $u$ such that $e_v \notin E_{out}$ and $f_v \notin E(A_{in})$.

As $T_P \neq T_Q$ there is a terminal $t$ contained in one of the two sets but not the other. Consider the case when $t \in T_Q \setminus T_P$, i.e., there is a $v \rightarrow t$ path in $A_{out}$ but no $t \rightarrow u$ path in $A_{in}$. The latter implies that $e_v$ cannot be reached from $t$ in $A_{in}$, as the $u \rightarrow v$ path $I$ contains no branching point of $A_{in}$. The in-arborescence $A_{in}$ does, however, contain a $t \rightarrow r$ path from $t$ to the root $r$. Since $e_v \notin E(A_{out})$, this means that the root $r$ can be reached from $v$ through the $v \rightarrow t$ path of $A_{out}$ and the $t \rightarrow r$ path without passing through $e_v$. Hence $e_v$ can safely be removed without making the solution $M$ infeasible. This contradicts the minimality of $M$.

In the case $t \in T_P \setminus T_Q$ a symmetric argument shows that the edge $f_v$ is redundant in $M$, which again contradicts its minimality. We have thus shown that $P$ and $Q$ are the only essential paths that meet in any endpoint of a shared path in the $W_M$-component $H$. Hence $H$ consists of exactly these two paths $P$ and $Q$, and the claim follows. \[\square\]

Consider the case when there is at most one shared path of $M$ that intersects with a $W_M$-component $H$. Since by Claim 2.4, $H$ consists of at most two essential paths, it is easy to see that in this case $H$ is a tree, and thus its treewidth is 1. If at least two shared paths of $M$ intersect with $H$, by Claim 2.4, $H$ contains edges of two

---

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
essential paths \( P \) and \( Q \) of \( A_{\text{in}} \) and \( A_{\text{out}} \) respectively. To show that in this case the treewidth of \( H \) is at most 2, we need the following observation on \( P \) and \( Q \).

Claim 2.5. Let \( I_1, \ldots, I_h \) be the connected components in the intersection of \( P \) and \( Q \), ordered in the way that \( P \) visits them, i.e., for any \( i \in \{1, \ldots, h-1\} \) there is a subpath of \( P \) with prefix \( I_i \), and suffix \( I_{i+1} \). The path \( Q \) visits the shared paths in the opposite order, i.e., for any \( i \in \{1, \ldots, h-1\} \) there is a subpath of \( Q \) with prefix \( I_i \) and suffix \( I_{i+1} \).

Proof. Assume this is not the case, so that there is an index \( i \in \{1, \ldots, h-1\} \) such that both \( P \) and \( Q \) contain subpaths with prefix \( I_i \) and suffix \( I_{i+1} \). This means that there are edges \( e \in E(P) \setminus E(Q) \) and \( f \in E(Q) \setminus E(P) \) which share the last vertex \( u \) of \( I_i \). Hence \( Q \) contains a \( u \rightarrow v \) subpath \( Q' \) to the first vertex \( v \) of \( I_{i+1} \), which does not contain the edge \( e \), and also \( P \) contains a \( u \rightarrow v \) subpath \( P' \), which does not contain the edge \( f \). As \( Q \) and therefore also \( Q' \), contains no branching point of \( A_{\text{out}} \), any terminal reachable from \( u \) through \( Q' \) in \( A_{\text{out}} \) is also reachable from \( u \) via the detour \( P' \). We can therefore remove edge \( f \in E(A_{\text{out}}) \) without violating the feasibility of \( M \). This however contradicts its minimality.

Claim 2.5 implies that the structure of \( H \) is roughly as shown in Figure 1 (the black edges shown in Figure 1 correspond to paths of length 0 or more, while the light and dotted edges correspond to paths of length at least 1). If we contract each path of length at least 1 to a path of length 1, then the resulting graph is planar and all vertices belong to the outer face. Such graphs are called outerplanar graphs. In other words, \( H \) is a subdivision of an outerplanar graph. Lemma B.4 shows that treewidth of subdivisions of outerplanar graphs is at most 2, which proves Lemma 2.1.

3. \( W[1] \)-hardness for SCSS in planar graphs. The goal of this section is to prove Theorem 1.2. We reduce from the Grid Tiling problem\(^5\) introduced by Marx [56]:

\[ k \times k \text{ Grid Tiling} \]

**Input:** Integers \( k \), \( n \), and \( k^2 \) nonempty sets \( S_{i,j} \subseteq [n] \times [n] \), where \( 1 \leq i, j \leq k \)

**Question:** For each \( 1 \leq i, j \leq k \) does there exist an entry \( \gamma_{i,j} \in S_{i,j} \) such that
- if \( \gamma_{i,j} = (x, y) \) and \( \gamma_{i,j+1} = (x', y') \), then \( x = x' \);
- if \( \gamma_{i,j} = (x, y) \) and \( \gamma_{i+1,j} = (x', y') \), then \( y = y' \).

\(^5\)The Grid Tiling problem has been defined in two (symmetrical) ways in the literature: either the first coordinate or the second coordinate remains the same in a row. Here, we follow the notation of [56], but the other definition also appears in some places (e.g., [23]).
Under ETH [45, 46], it was shown by Chen et al. [15] that \(k\)-CLIQUE\(^6\) does not admit an algorithm running in time \(f(k) \cdot n^{o(k)}\) for any computable function \(f\). There is a simple reduction [23, Theorem 14.28] from \(k\)-CLIQUE to \(k \times k\) GRID TILING implying the same runtime lower bound for the latter problem. To prove Theorem 1.2, we give a reduction which transforms the problem of \(k \times k\) GRID TILING into a planar instance of SCSS with \(O(k^2)\) terminals. We design two types of gadgets: the connector gadget and the main gadget. The reduction from GRID TILING represents each cell of the grid with a copy of the main gadget, with a connector gadget between main gadgets that are adjacent either horizontally or vertically (see Figure 2).

The proof of Theorem 1.2 is divided into the following steps: in section 3.1, we first introduce the connector gadget and Lemma 3.3 states the existence of a particular type of connector gadget. In section 3.2, we introduce the main gadget and Lemma 3.6 states the existence of a particular type of main gadget. Section 3.3 describes the construction of the planar instance \((G^*, T^*)\) of SCSS. The two directions implying the reduction from GRID TILING to SCSS are proved in sections 3.4 and 3.5, respectively.

\(^6\)The \(k\)-CLIQUE problem asks whether there is a clique of size \(\geq k\).
Using Lemmas 3.3 and 3.6 as a black box, we prove Theorem 1.2 in section 3.6. The proofs of Lemmas 3.3 and 3.6 are given in sections 4 and 5, respectively.

3.1. Existence of connector gadgets. A connector gadget \( CG_n \) is a directed (embedded) planar graph with \( O(n^2) \) vertices and positive integer weights\(^7\) on its edges. It has a total of \( 2n + 2 \) distinguished vertices divided into the following 3 types:

- The vertices \( p, q \) are called internal-distinguished vertices.
- The vertices \( p_1, p_2, \ldots, p_n \) are called source-distinguished vertices.
- The vertices \( q_1, q_2, \ldots, q_n \) are called sink-distinguished vertices.

Let \( P = \{p_1, p_2, \ldots, p_n\} \) and \( Q = \{q_1, q_2, \ldots, q_n\} \). The vertices \( P \cup Q \) appear in the clockwise order \( p_1, p_2, \ldots, p_n, q_n, \ldots, q_1 \) on the boundary of the gadget. In the connector gadget \( CG_n \), every vertex in \( P \) is a source and has exactly one outgoing edge. Also every vertex in \( Q \) is a sink and has exactly one incoming edge.

**Definition 3.1.** An edge set \( E' \subseteq E(CG_n) \) satisfies the connectedness property if each of the following four conditions hold for the graph \( CG_n[E'] \):

1. \( p \) can be reached from some vertex in \( P \);
2. \( q \) can be reached from some vertex in \( P \);
3. \( p \) can reach some vertex in \( Q \);
4. \( q \) can reach some vertex in \( Q \).

**Definition 3.2.** An edge set \( E' \) satisfying the connectedness property represents an integer \( i \in [n] \) if in \( E' \) the only outgoing edge from \( P \) is the one incident to \( p_i \) and the only incoming edge into \( Q \) is the one incident to \( q_i \).

The next lemma shows we can construct a particular type of connector gadget.

**Lemma 3.3.** Given an integer \( n \) one can construct in polynomial time a connector gadget \( CG_n \) and an integer \( C^*_n \) such that the following two properties hold:\(^8\)

1. For every \( i \in [n] \), there is an edge set \( E_i \subseteq E(CG_n) \) of weight \( C^*_n \) such that \( E_i \) satisfies the connectedness property and represents \( i \). Note that, in particular, \( E_i \) contains a \( p_i \sim q_i \) path (via \( p \) or \( q \)).
2. If there is an edge set \( E' \subseteq E(CG_n) \) such that \( E' \) has weight at most \( C^*_n \) and \( E' \) satisfies the connectedness property, then \( E' \) has weight exactly \( C^*_n \) and it represents some \( i \in [n] \).

3.2. Existence of main gadgets. A main gadget \( MG \) is a directed (embedded) planar graph with \( O(n^3) \) vertices and positive integer weights on its edges. It has \( 4n \) distinguished vertices given by the following four sets:

- The set \( L = \{\ell_1, \ell_2, \ldots, \ell_n\} \) of left-distinguished vertices.
- The set \( R = \{r_1, r_2, \ldots, r_n\} \) of right-distinguished vertices.
- The set \( T = \{t_1, t_2, \ldots, t_n\} \) of top-distinguished vertices.
- The set \( B = \{b_1, b_2, \ldots, b_n\} \) of bottom-distinguished vertices.

The distinguished vertices appear in the (clockwise) order \( t_1, \ldots, t_n, r_1, \ldots, r_n, b_1, \ldots, b_n, \ell_1, \ldots, \ell_n \) on the boundary of the gadget. In the main gadget \( MG \), every vertex in \( L \cup T \) is a source and has exactly one outgoing edge. Also each vertex in \( R \cup B \) is a sink and has exactly one incoming edge.

**Definition 3.4.** An edge set \( E' \subseteq E(MG) \) satisfies the connectedness property if each of the following four conditions hold for the graph \( MG[E'] \):

\(^7\)Weights are polynomial in \( n \).
\(^8\)We use the notation \( C^*_n \) to emphasize that \( C^* \) depends only on \( n \).
1. There is a directed path from some vertex in \( L \) to \( R \cup B \).
2. There is a directed path from some vertex in \( T \) to \( R \cup B \).
3. Some vertex in \( R \) can be reached from \( L \cup T \).
4. Some vertex in \( B \) can be reached from \( L \cup T \).

**Definition 3.5.** An edge set \( E' \subseteq E(MG) \) represents a pair \((i, j) \in [n] \times [n]\) if each of the following five conditions holds:
- The only edge of \( E' \) leaving \( L \) is the one incident to \( \ell_i \).
- The only edge of \( E' \) entering \( R \) is the one incident to \( r_i \).
- The only edge of \( E' \) leaving \( T \) is the one incident to \( t_j \).
- The only edge of \( E' \) entering \( B \) is the one incident to \( b_j \).
- \( E' \) contains an \( \ell_i \rightarrow r_i \) path and an \( t_j \rightarrow b_j \) path.

The next lemma shows we can construct a particular type of main gadget.

**Lemma 3.6.** Given a subset \( S \subseteq [n] \times [n] \), one can construct in polynomial time a main gadget \( MG \) and an integer \( M^*_n \) such that the following two properties hold: \(^9\)

1. For every \((i, j) \in S\) there is an edge set \( E_{i,j} \subseteq E(MG_S) \) of weight \( M^*_n \) such that \( E_{i,j} \) represents \((i, j)\). Note that the last condition of Definition 3.5 implies that \( E_{i,j} \) satisfies the connectedness property.
2. If there is an edge set \( E' \subseteq E(MG_S) \) such that \( E' \) has weight at most \( M^*_n \) and satisfies the connectedness property, then \( E' \) has weight exactly \( M^*_n \) and represents some \((i, j) \in S\).

### 3.3. Construction of the SCSS instance

In order to prove Theorem 1.2, we reduce from the Grid Tiling problem. The following assumption will be helpful in handling some of the border cases of the gadget construction. We may assume that \( 1 < x, y < n \) holds for every \((x, y) \in S_{i,j}\): indeed, we can increase \( n \) by two and replace every \((x, y) \) by \((x + 1, y + 1)\) without changing the problem. This is just a minor technical modification \(^{10}\) which is introduced to make some of the arguments in section 5 cleaner.

Given an instance \((k, n, \{S_{i,j} : i, j \in [k]\})\) of Grid Tiling, we construct an instance \((G^*, T^*)\) of SCSS in the following way (see Figure 2):

- We introduce a total of \( k^2 \) main gadgets and \( 2k(k+1) \) connector gadgets.
- For every set \( S_{i,j} \) in the Grid Tiling instance, we construct a main gadget \( MG_{i,j} \) using Lemma 3.6 for the subset \( S_{i,j} \).
- Half of the connector gadgets have the same orientation as in Figure 3 (with the \( p_i \) vertices on the top side and the \( q_i \) vertices on the bottom side), and we call them HCG to denote horizontal connector gadgets. \(^{11}\) The other half of the connector gadgets are rotated anticlockwise by 90 degrees with respect to the orientation of Figure 3, and we call them VCG to denote vertical connector gadgets. The internal-distinguished vertices of the connector gadgets are shown in Figure 2.
- For each \( 1 \leq i, j \leq k \), the main gadget \( MG_{i,j} \) is surrounded by the following four connector gadgets:

\(^9\) We use the notation \( M^*_n \) to emphasize that \( M^* \) depends only on \( n \), and not on the set \( S \).

\(^{10}\) For the interested reader, what this modification does is to ensure no shortcut edge added in section 5.1 has either endpoint on the unbounded face of the planar embedding of the main gadget provided in Figure 4. This helps to streamline the proofs by avoiding the need to have to consider any special cases.

\(^{11}\) The horizontal connector gadgets are so called because they connect things horizontally as seen by the reader.
Fig. 3. The connector gadget for $n = 3$. A set of edges representing 3 is highlighted in the figure.

1. The vertical connector gadget $VCG_{i,j}$ is on the top and $VCG_{i+1,j}$ is on the bottom. Identify (or glue together) each sink-distinguished vertex of $VCG_{i,j}$ with the top-distinguished vertex of $MCG_{i,j}$ of the same index. Similarly identify each source-distinguished vertex of $VCG_{i+1,j}$ with the bottom-distinguished vertex of $MCG_{i,j}$ of the same index.

2. The horizontal connector gadget $HCG_{i,j}$ is on the left and $HCG_{i,j+1}$ is on the right. Identify (or glue together) each sink-distinguished vertex of $HCG_{i,j}$ with the left-distinguished vertex of $MCG_{i,j}$ of the same index.
Similarly, identify each source-distinguished vertex of $HCG_{i,j+1}$ with the right-distinguished vertex of $MCG_{i,j}$ of the same index.

- We introduce two special vertices $x^*, y^*$ and add an edge $(x^*, y^*)$ of weight 0.
- For each $1 \leq i \leq k$, add an edge of weight 0 from $y^*$ to each source-distinguished vertex of the vertical connector gadget $VCG_{1,i}$.
- For each $1 \leq j \leq k$, add an edge of weight 0 from $y^*$ to each source-distinguished vertex of the horizontal connector gadget $HCG_{j,1}$.
- For each $1 \leq i \leq k$, add an edge of weight 0 from each sink-distinguished vertex of the vertical connector gadget $VCG_{k+1,i}$ to $x^*$.
- For each $1 \leq j \leq k$, add an edge of weight 0 from each sink-distinguished vertex of the horizontal connector gadget $HCG_{j,k+1}$ to $x^*$.
- For each $i \in [k], j \in [k+1]$, denote the two internal-distinguished vertices of $HCG_{i,j}$ by $\{p_{i,j}^h, q_{i,j}^h\}$.
- For each $i \in [k+1], j \in [k]$, denote the two internal-distinguished vertices of $VCG_{i,j}$ by $\{p_{i,j}^v, q_{i,j}^v\}$.
- The set of terminals $T^*$ for the SCSS instance on $G^*$ is $\{x^*, y^*\} \cup \{p_{i,j}^h, q_{i,j}^h \mid 1 \leq i \leq k + 1, 1 \leq j \leq k\} \cup \{p_{i,j}^v, q_{i,j}^v \mid 1 \leq i \leq k, 1 \leq j \leq k + 1\}$.
- We note that the total number of terminals is $|T^*| = 4k(k + 1) + 2 = O(k^2)$.
- The edge set of $G^*$ is a disjoint union of
  - edges of main gadgets;
  - edges of horizontal connector gadgets;
  - edges of vertical connector gadgets;
  - edges from $y^*$ to source-distinguished vertices of the vertical connector gadgets $VCG_{1,i}$ (for each $i \in [k]$), and from $y^*$ to source-distinguished vertices of horizontal connector gadgets $HCG_{j,1}$ (for each $j \in [k]$);
  - edges from sink-distinguished vertices of the vertical connector gadgets $VCG_{k+1,i}$ (for each $i \in [k]$) to $x^*$, and from sink-distinguished vertices of horizontal connector gadgets $HCG_{j,k+1}$ (for each $j \in [k]$) to $x^*$;
  - the edge $(x^*, y^*)$.

Define the following quantity:

$$W^*_n = k^2 \cdot M^*_n + 2k(k + 1) \cdot C^*_n.$$  

In the next two sections, we show that Grid Tiling has a solution if and only if the SCSS instance $(G^*, T^*)$ has a solution of weight at most $W^*_n$.

### 3.4. Grid Tiling has a solution $\Rightarrow$ SCSS has a solution of weight $\leq W^*_n$.

**Lemma 3.7.** If the grid tiling instance $(k, n, \{S_{i,j} : i, j \in [k]\})$ has a solution, then the SCSS instance $(G^*, T^*)$ has a solution of weight at most $W^*_n$.

**Proof.** Since Grid Tiling has a solution, for each $1 \leq i, j \leq k$ there is an entry $(x_{i,j}, y_{i,j}) = \gamma_{i,j} \in S_{i,j}$ such that
- for every $i \in [k]$, we have $x_{i,1} = x_{i,2} = \cdots = x_{i,k} = \alpha_i$;
- for every $j \in [k]$, we have $y_{1,j} = y_{2,j} = \cdots = y_{k,j} = \beta_j$.

We build a solution $E^*$ for the SCSS instance $(G^*, T^*)$ and show that it has weight at most $W^*_n$. In the edge set $E^*$, we take the following edges:

1. The edge $(x^*, y^*)$ which has weight 0.
2. For each $j \in [k]$ the edge of weight 0 from $y^*$ to the source-distinguished vertex of $VCG_{1,j}$ of index $\beta_j$, and the edge of weight 0 from the sink-distinguished vertex of $VCG_{k+1,j}$ of index $\beta_j$ to $x^*$.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
3. For each $i \in [k]$ the edge of weight 0 from $y^*$ to the source-distinguished vertex of $HCG_{i,1}$ of index $\alpha_i$, and the edge of weight 0 from the sink-distinguished vertex of $HCG_{i,k+1}$ of index $\alpha_i$ to $x^*$.

4. For each $1 \leq i, j \leq k$ for the main gadget $MG_{i,j}$, use Lemma 3.6(1) to generate a solution $E^M_{i,j}$ which has weight $M^*_{i,j}$ and represents $(\alpha_i, \beta_j)$.

5. For each $1 \leq i \leq k$ and $1 \leq j \leq k + 1$ for the horizontal connector gadget $HCG_{i,j}$, use Lemma 3.3(1) to generate a solution $E^H_{i,j}$ of weight $C^*_n$ which represents $\alpha_i$.

6. For each $1 \leq j \leq k$ and $1 \leq i \leq k + 1$ for the vertical connector gadget $VCG_{i,j}$, use Lemma 3.3(1) to generate a solution $E^V_{i,j}$ of weight $C^*_n$ which represents $\beta_j$.

The weight of $E^*$ is $k^2 \cdot M^*_n + k(k + 1) \cdot C^*_n + k(k + 1) \cdot C^*_n = W^*_n$. It remains to show that $E^*$ is a solution for the SCSS instance $(G^*, T^*)$. Since we have already picked up the edge $(x^*, y^*)$, it is enough to show that for any terminal $t \in T^* \setminus \{x^*, y^*\}$, both $t \rightsquigarrow x^*$ and $y^* \rightsquigarrow t$ paths exist in $E^*$. Then for any two terminals $t_1, t_2$, there is a $t_1 \rightsquigarrow t_2$ path in $G^*$.

We now show the existence of both a $t \rightsquigarrow x^*$ path and a $y^* \rightsquigarrow t$ path in $E^*$ when $t$ is a terminal in a vertical connector gadget. Without loss of generality, let $t$ be the terminal $p^i_{i,j}$ for some $1 \leq i \leq k, 1 \leq j \leq k + 1$.

- **Existence of $p^i_{i,j} \rightsquigarrow x^*$ path in $E^*$:** By Lemma 3.3(1), the terminal $p^i_{i,j}$ can reach the sink-distinguished vertex of $VCG_{i,j}$ which has the index $\beta_j$. This vertex is the top-distinguished vertex of the index $\beta_j$ of the main gadget $MG_{i,j}$. By Definition 3.5, there is a path from this vertex to the bottom-distinguished vertex of the index $\beta_j$ of the main gadget $MG_{i,j}$. However, this vertex is exactly the same vertex as the sink-distinguished vertex of the index $\beta_j$ of $VCG_{i,j+1}$. By Lemma 3.3(1), the source-distinguished vertex of the index $\beta_j$ of $VCG_{i,j+1}$ can reach the sink-distinguished vertex of the index $\beta_j$ of $VCG_{i,j+1}$. This vertex is exactly the top-distinguished vertex of $MG_{i,j+1}$. Continuing in this way we can reach the source-distinguished vertex of the index $\beta_j$ of $VCG_{k+1,j}$. By Lemma 3.3(1), this vertex can reach the sink-distinguished vertex of the index $\beta_j$ of $VCG_{k+1,j}$. But $E^*$ also contains an edge of weight 0 from this sink-distinguished vertex to $x^*$, and hence there is a $p^i_{i,j} \rightsquigarrow x^*$ path in $E^*$.

- **Existence of $y^* \rightsquigarrow p^i_{i,j}$ path in $E^*$:** Recall that $E^*$ contains an edge of weight 0 from $y^*$ to the source-distinguished vertex of the index $\beta_j$ of $VCG_{i,j}$. If $i = 1$, then by Lemma 3.3(1), there is a path from this vertex to $p^1_{1,j}$. If $i \geq 2$, then by Lemma 3.3(1), there is a path from the source-distinguished vertex of the index $\beta_j$ of $VCG_{1,j}$ to the sink-distinguished vertex of the index $\beta_j$ of $VCG_{1,j}$. But this is the top-distinguished vertex of $MG_{1,j}$ of the index $\beta_j$. By Definition 3.5, from this vertex we can reach the bottom-distinguished vertex of the index $\beta_j$ of $MG_{1,j}$. However, this vertex is exactly the source-distinguished vertex of index $\beta_j$ of $VCG_{2,j}$. Continuing this way we can reach the source-distinguished vertex of the index $\beta_j$ of $VCG_{i,j}$. By Lemma 3.3(1), from this vertex we can reach $p^i_{i,j}$. Hence there is a $y^* \rightsquigarrow p^i_{i,j}$ path in $E^*$.

The arguments when $t$ is a terminal in a horizontal connector gadget are similar, and we omit the details here.

**3.5. SCSS has a solution of weight $\leq W^*_n \Rightarrow GRID TILING has a solution.** First we show the following preliminary claim.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
CLAIM 3.8. Let \( E' \) be any solution to the SCSS instance \((G^*, T^*)\). Then

- \( E' \) restricted to each connector gadget satisfies the connectedness property (see Definition 3.1);
- \( E' \) restricted to each main gadget satisfies the connectedness property (see Definition 3.4).

Proof. First we show that the edge set \( E' \) restricted to each connector gadget satisfies the connectedness property. Consider a horizontal connector gadget \( HCG_{i,j} \) for some \( 1 \leq j \leq k + 1, 1 \leq i \leq k \). This gadget contains two terminals: \( p_{i,j}^h \) and \( q_{i,j}^h \). The only incoming edges from \( G^* \setminus HCG_{i,j} \) into \( HCG_{i,j} \) are incident onto the source-distinguished vertices of \( HCG_{i,j} \), and the only outgoing edges from \( HCG_{i,j} \) into \( G^* \setminus HCG_{i,j} \) are incident on the sink-distinguished vertices of \( HCG_{i,j} \). Since \( E' \) is a solution of the SCSS instance \((G^*, T^*)\) it follows that \( E' \) contains a path from \( p_{i,j}^h \) to the terminals in \( T^* \setminus \{p_{i,j}^s \cup q_{i,j}^b\} \). Since the only outgoing edges from \( HCG_{i,j} \) into \( G^* \setminus HCG_{i,j} \) are incident on the sink-distinguished vertices of \( HCG_{i,j} \) it follows that \( p_{i,j}^h \) can reach some sink-distinguished vertex of \( HCG_{i,j} \) in the solution \( E' \). The other three conditions of Definition 3.1 can be verified similarly, and hence \( E' \) restricted to each main connector satisfies the connectedness property.

Next we argue that \( E' \) restricted to each main gadget satisfies the connectedness property. Consider a main gadget \( MG_{i,j} \). Since \( E' \) is a solution for the SCSS instance \((G^*, T^*)\) it follows that the terminal \( p_{i,j}^m \) from \( HCG_{i,j} \) is able to reach other terminals of \( T^* \). However, the only outgoing edges from \( HCG_{i,j} \) into \( G^* \setminus HCG_{i,j} \) are incident on the sink-distinguished vertices of \( HCG_{i,j} \). Moreover, each sink-distinguished vertex of \( HCG_{i,j} \) is identified with a left-distinguished vertex of \( MG_{i,j} \) of the same index. Hence, these outward paths from \( p_{i,j}^m \) to other terminals of \( T^* \) must continue through the left-distinguished vertices of \( MG_{i,j} \). However, the only outgoing edges from \( MG_{i,j} \) into \( G^* \setminus MG_{i,j} \) are incident on the right-distinguished vertices or bottom-distinguished vertices of \( HCG_{i,j} \). Hence, some left-distinguished vertex of \( MG_{i,j} \) can reach some vertex in the set given by the union of right-distinguished and bottom-distinguished vertices of \( MG_{i,j} \). Hence the first condition of Definition 3.4 is satisfied. Similarly it can be shown the other three conditions of Definition 3.4 also hold, and hence \( E' \) restricted to each main gadget satisfies the connectedness property.

Now we are ready to prove the following lemma.

LEMMA 3.9. If the SCSS instance \((G^*, T^*)\) has a solution \( E'' \) of weight at most \( W_n^* \), then the Grid Tiling instance \((k, n, \{S_{i,j} : i, j \in [k]\})\) has a solution.

Proof. By Claim 3.8, the edge set \( E'' \) restricted to any connector gadget satisfies the connectedness property and the edge set \( E'' \) restricted to any main gadget satisfies the connectedness property. Let \( C \) and \( M \) be the sets of connector and main gadgets, respectively. Recall that \( |C| = 2k(k + 1) \) and \( |M| = k^2 \). Recall that we have defined \( W_n^* = k^2 \cdot M_n^* + 2k(k + 1)C_n^* \). Let \( C' \subseteq C \) be the set of connector gadgets that have weight at most \( C_n^* \) in \( E'' \). By Lemma 3.3(2), each connector gadget from the set \( C' \) has weight exactly \( C_n^* \). Since all edge weights in connector gadgets are positive integers, each connector gadget from the set \( C \setminus C' \) has weight at least \( C_n^* + 1 \). Similarly, let \( M' \subseteq M \) be the set of main gadgets which have weight at most \( M_n^* \) in \( E'' \). By Lemma 3.6(2), each main gadget from the set \( M' \) has weight exactly \( M_n^* \). Since all edge weights in main gadgets are positive integers, each main gadget from the set \( M \setminus M' \) has weight at least \( M_n^* + 1 \). As any two gadgets are pairwise edge disjoint,
we have

\[ W_n^* = k^2 \cdot M_n^* + 2k(k + 1)C_n^* \geq |\mathcal{M} \setminus \mathcal{M}'| \cdot (M_n^* + 1) + |\mathcal{M}'| \cdot M_n^* + |\mathcal{C} \setminus \mathcal{C}'| \cdot (C_n^* + 1) + |\mathcal{C}'| \cdot C_n^* \]

\[ = |\mathcal{M}| \cdot M_n^* + |\mathcal{C}| \cdot C_n^* + |\mathcal{M} \setminus \mathcal{M}'| + |\mathcal{C} \setminus \mathcal{C}'| \]

\[ = k^2 \cdot M_n^* + 2k(k + 1)C_n^* + |\mathcal{M} \setminus \mathcal{M}'| + |\mathcal{C} \setminus \mathcal{C}'| \]

\[ = W_n^* + |\mathcal{M} \setminus \mathcal{M}'| + |\mathcal{C} \setminus \mathcal{C}'|. \]

This implies \( |\mathcal{M} \setminus \mathcal{M}'| = 0 = |\mathcal{C} \setminus \mathcal{C}'|. \) However, we had \( M' \subseteq \mathcal{M} \) and \( C' \subseteq \mathcal{C}. \) Therefore, \( M' = \mathcal{M} \) and \( C' = \mathcal{C}. \) Hence in \( E^n, \) each connector gadget has weight \( C_n^* \) and each main gadget has weight \( M_n^* \). From Lemma 3.3(2) and Lemma 3.6(2), we have

- for each vertical connector gadget \( VCG_{i,j}, \) the restriction of the edge set \( E'' \) to \( VCG_{i,j} \) represents an integer \( \beta_{i,j} \) where \( i \in [k + 1], j \in [k]; \)
- for each horizontal connector gadget \( HCG_{i,j}, \) the restriction of the edge set \( E'' \) to \( HCG_{i,j} \) represents an integer \( \alpha_{i,j} \) where \( i \in [k], j \in [k + 1]; \)
- for each main gadget \( MG_{i,j}, \) the restriction of the edge set \( E'' \) to \( MG_{i,j} \) represents an ordered pair \( (\alpha'_{i,j}, \beta'_{i,j}) \in S_{i,j}, \) where \( i, j \in [k]. \)

Consider the main gadget \( MG_{i,j} \) for any \( 1 \leq i, j \leq k. \) We can make the following observations:

- \( \beta_{i,j} = \beta'_{i,j}: \) By Lemma 3.3(2) and Definition 3.2, the terminal vertices in \( VCG_{i,j} \) can exit the vertical connector gadget only via the unique edge entering the sink-distinguished vertex of index \( \beta_{i,j}. \) By Lemma 3.6(2) and Definition 3.5, the only edge in \( E'' \) incident to any top-distinguished vertex of \( MG_{i,j} \) is the unique edge leaving the top-distinguished vertex of the index \( \beta'_{i,j}. \) Hence if \( \beta_{i,j} \neq \beta'_{i,j} \) then the terminals in \( VCG_{i,j} \) will not be able to exit \( VCG_{i,j} \) and reach other terminals.
- \( \beta'_{i,j} = \beta_{i+1,j}: \) By Lemma 3.3(2) and Definition 3.2, the unique edge entering \( VCG_{i+1,j} \) is the edge entering the source-distinguished vertex of the index \( \beta_{i+1,j}. \) By Lemma 3.6(2) and Definition 3.5, the only edge in \( E'' \) incident to any bottom-distinguished vertex of \( MG_{i,j} \) is the unique edge entering the bottom-distinguished vertex of index \( \beta'_{i,j}. \) Hence if \( \beta'_{i,j} \neq \beta_{i+1,j} \) then the terminals in \( VCG_{i+1,j} \) cannot be reached from the other terminals.
- \( \alpha_{i,j} = \alpha'_{i,j}: \) By Lemma 3.3(2) and Definition 3.2, the paths starting at the terminal vertices in \( HCG_{i,j} \) can leave the horizontal connector gadget only via the unique edge entering the sink-distinguished vertex of index \( \alpha_{i,j}. \) By Lemma 3.6(2) and Definition 3.5, the only edge in \( E'' \) incident to any left-distinguished vertex of \( MG_{i,j} \) is the unique edge leaving the left-distinguished vertex of the index \( \alpha'_{i,j}. \) Hence if \( \alpha_{i,j} \neq \alpha'_{i,j} \) then the terminals in \( HCG_{i,j} \) will not be able to reach other terminals.
- \( \alpha'_{i,j} = \alpha_{i,j+1}: \) By Lemma 3.3(2) and Definition 3.2, the unique edge entering \( HCG_{i,j+1} \) is the edge entering the source-distinguished vertex of index \( \alpha_{i,j+1}. \) By Lemma 3.6(2) and Definition 3.5, the only edge in \( E'' \) incident to any right-distinguished vertex of \( MG_{i,j} \) is the unique edge entering the right-distinguished vertex of index \( \alpha'_{i,j}. \) Hence if \( \alpha'_{i,j} \neq \alpha_{i,j+1} \) then the terminals in \( HCG_{i,j+1} \) cannot be reached from the other terminals.

We claim that for \( 1 \leq i, j \leq k, \) the entries \( (\alpha'_{i,j}, \beta'_{i,j}) \in S_{i,j} \) form a solution for the Grid Tiling instance. For this we need to check two conditions:
TIGHT BOUNDS FOR PLANAR SCSS (AND EXTENSIONS) 337

- $\alpha''_{i,j} = \alpha''_{i,j+1}$: This holds because $\alpha_{i,j} = \alpha'_{i,j} = \alpha_{i,j+1} = \alpha''_{i,j+1}$.
- $\beta''_{i,j} = \beta''_{i+1,j}$: This holds because $\beta_{i,j} = \beta'_{i,j} = \beta_{i+1,j} = \beta''_{i+1,j}$.

This completes the proof of the lemma.

3.6. Proof of Theorem 1.2. Finally we are ready to prove Theorem 1.2 which is restated below.

Theorem 1.2. The edge-unweighted version of the SCSS problem is $W[1]$-hard parameterized by the number of terminals $k$, even when the underlying directed graph is planar. Moreover, under ETH, the SCSS problem on planar graphs cannot be solved in $f(k) \cdot n^\omega(\sqrt{k})$ time, where $f$ is any computable function, $k$ is the number of terminals, and $n$ is the number of vertices in the instance.

Proof. Each connector gadget has $O(n^2)$ vertices and $G^*$ has $O(k^2)$ connector gadgets. Each main gadget has $O(n^3)$ vertices and $G^*$ has $O(k^2)$ main gadgets. It is easy to see that the graph $G^*$ has $O(n^3k^2) = poly(n,k)$ vertices. Moreover, the graph $G^*$ can be constructed in $poly(n + k)$ time: recall that each connector gadget (Lemma 3.3) and main gadget (Lemma 3.6) can be constructed in polynomial time. Each main gadget and connector gadget is planar, and any two gadgets are pairwise edge disjoint. Moreover, the 0-weight edges incident on $x^*$ or $y^*$ do not affect planarity (see Figure 2 for a planar embedding). Hence, $G^*$ is planar.

It is known [23, Theorem 14.28] that $k \times k$ GRID TILING is $W[1]$-hard parameterized by $k$, and under ETH cannot be solved in $f(k) \cdot n^\omega(k)$ for any computable function $f$. Combining the two directions from sections 3.4 and 3.5, we get a parameterized reduction from $k \times k$ GRID TILING to a planar instance of SCSS with $O(k^2)$ terminals. Hence, it follows that SCSS on planar graphs is $W[1]$-hard and under ETH cannot be solved in $f(k) \cdot n^\omega(\sqrt{k})$ time for any computable function $f$.

This shows that the $2^{O(k)} \cdot n^{O(\sqrt{k})}$ algorithm for SCSS on planar graphs given in Theorem 1.1 is asymptotically optimal.

4. Proof of Lemma 3.3: Constructing the connector gadget. We prove Lemma 3.3 in this section, by constructing a connector gadget satisfying the specifications of section 3.1.

4.1. Different types of edges in the connector gadget. Before proving Lemma 3.3, we first describe the construction of the connector gadget in more detail (see Figure 3). The connector gadget has $2n + 4$ rows denoted by $R_0, R_1, R_2, \ldots, R_{2n+3}$ and $4n + 1$ columns denoted by $C_0, C_1, \ldots, C_{4n}$. Let us denote the vertex at the intersection of row $R_i$ and column $C_j$ by $v^i_j$. We now describe the different kinds of edges present in the connector gadget.

1. Source edges: For each $i \in [n]$, there is an edge $(p_i, v^{2i+1}_0)$. These edges are together called source edges.

2. Sink edges: For each $i \in [n]$, there is an edge $(v^{2n+2i+1}_{2n+3}, q_i)$. These edges are together called sink edges.

3. Terminal edges: The union of the sets of edges incident to the terminals $p$ or $q$ are called terminal edges. The set of edges incident on $p$ is $\{(p, v^{2i+1}_{2n+3} : i \in [n])\} \cup \{(v^{2i}_0, p : i \in [n])\}$. The set of edges incident on $q$ is $\{(q, v^{2i+1}_{2n+3} : i \in [n])\} \cup \{(v^{2i}_0, q : i \in [n])\}$.

4. Inrow edges:
   - Inrow up edges: For each $0 \leq i \leq n + 1$, we call the $\uparrow$ edges connecting vertices of row $R_{2i+1}$ to $R_{2i}$ as inrow up edges. Explicitly, this set of edges is given by $\{(v^{2i+1}_j, v^{2i}_j) : 0 \leq j \leq 2n\}$.
\textbullet{} Inrow down edges: For each \(0 \leq i \leq n+1\), we call the \(\downarrow\) edges connecting vertices of row \(R_i\) to \(R_{i+1}\) as inrow down edges. Explicitly, this set of edges is given by \(\{(v_{2i+1}^{2j-1}, v_{2i+1}^{2j-1}) \mid 1 \leq j \leq 2n\}\).

\textbullet{} Inrow left edges: For each \(0 \leq i \leq 2n+3\), we call the \(\leftarrow\) edges connecting vertices of row \(R_i\) as inrow left edges. We explicitly list the set of inrow left edges for even-numbered and odd-numbered rows below:

- For each \(0 \leq i \leq n+1\), the set of inrow left edges for the row \(R_{2i}\) is given by \(\{(v_{2i}^{2j-1}, v_{2i}^{2j-1}) \mid j \in [2n]\}\).
- For each \(0 \leq i \leq n+1\), the set of inrow left edges for the row \(R_{2i+1}\) is given by \(\{(v_{2i+1}^{2j-1}, v_{2i+1}^{2j-1}) \mid j \in [2n]\}\).

\textbullet{} Inrow right edges: For each \(0 \leq i \leq 2n+3\), we call the \(\rightarrow\) edges connecting vertices of row \(R_i\) as inrow right edges. We explicitly list the set of inrow right edges for even-numbered and odd-numbered rows below:

- For each \(0 \leq i \leq n+1\), the set of inrow right edges for the row \(R_{2i}\) is given by \(\{(v_{2i}^{2j-2}, v_{2i}^{2j-2}) \mid j \in [2n]\}\).
- For each \(0 \leq i \leq n+1\), the set of inrow right edges for the row \(R_{2i+1}\) is given by \(\{(v_{2i+1}^{2j-2}, v_{2i+1}^{2j-2}) \mid j \in [2n]\}\).

\textbullet{} Interrow edges: For each \(i \in [n+1]\) and each \(j \in [2n]\), we subdivide the edge \((v_{2i+1}^{2j-1}, v_{2i+1}^{2j-1})\) by introducing a new vertex \(w_i^j\) and adding the edges \((v_{2i+1}^{2j-1}, w_i^j)\) and \((w_i^j, v_{2i+1}^{2j-1})\). All these edges are together called interrow edges. Note that there is a total of \(4n(n+1)\) interrow edges.

\textbullet{} Shortcuts: There are \(2n\) shortcut edges, namely, \(e_1, e_2, \ldots, e_n\) and \(f_1, f_2, \ldots, f_n\). They are drawn as follows:

- The edge \(e_i\) is given by \((v_{2i-2}^{2n-2i+2}, v_{n-i+1}^{2n+2i})\).
- The edge \(f_i\) is given by \((v_{n-i+1}^{2n-2i+2}, v_{2n-2i+3}^{2n+2i})\).

4.2. Assigning weights in the connector gadget. Fix the quantity \(B = 18n^2\). We assign weights to the edges as follows:

1. For \(i \in [n]\), the source edge \((p_i, v_{0}^{2i-1})\) has weight \(B^5 + (n - i + 1)\).
2. For \(i \in [n]\), the sink edge \((v_{2n+3i-1}^{2n+3i-1}, q_i)\) has weight \(B^3 + i\).
3. Each terminal edge has weight \(B^4\).
4. Each inrow up edge has weight \(B^3\).
5. Each interrow edge has weight \(B^2/2\) each.
6. Each inrow right edge has weight \(B^3\).
7. For each \(i \in [n]\), the shortcut edge \(e_i\) has weight \(n \cdot i\).
8. For each \(j \in [n]\), the shortcut edge \(f_j\) has weight \(n(n - j + 1)\).
9. Each inrow left edge and inrow down edge has weight 0.

Now we define the quantity \(C_n^*\) given in the statement of Lemma 3.3:

\begin{equation}
C_n^* = 2B^5 + 4B^4 + 2(n + 1)B^3 + (n + 1)B^2 + (4n - 2)B + (n + 1)^2.
\end{equation}

In the next two sections, we prove the two statements of Lemma 3.3.

4.3. For every \(i \in [n]\), there is a solution \(E_i\) of weight \(C_n^*\) that satisfies the connectedness property and represents \(i\). Let \(E_i\) be the union of the following sets of edges:

- Select the edges \((p_i, v_{0}^{2i-1})\) and \((v_{2n+3i-1}^{2n+3i-1}, q_i)\). This incurs a weight of \(B^5 + (n - i + 1) + B^3 + i = 2B^5 + (n + 1)\).
- The two terminal edges \((p_i, v_{2n-2i+3}^{0})\) and \((v_{2n-2i+3}^{0}, p_i)\). This incurs a weight of \(2B^4\).
- The two terminal edges \((q, v_{2n-2i+3}^{n})\) and \((v_{2n-2i+3}^{n}, q)\). This incurs a weight of \(2B^4\).
• All 2n inrow right edges and 2n inrow left edges which occur between vertices of \(R_{2n-2i+2}\). This incurs a weight of \(2n \cdot B\) since each inrow left edge has weight 0 and each inrow right edge has weight \(B\).

• All 2n inrow right edges and 2n inrow left edges which occur between vertices of \(R_{2n-2i+3}\). This incurs a weight of \(2n \cdot B\) since each inrow left edge has weight 0 and each inrow right edge has weight \(B\).

• All the \(2n+1\) inrow up edges that are between vertices of \(R_{2n-2i+2}\) and \(R_{2n-2i+3}\). These edges are given by \((v_{2n-2i+3}, v_{2n-2i+2})\) for \(0 \leq j \leq 2n\). This incurs a weight of \((2n+1)B^3\).

• All 2n inrow down edges that occur between vertices of row \(R_{2n-2i+2}\) and row \(R_{2n-2i+3}\). This incurs a weight of 0, since each inrow down edge has weight 0.

• The vertically downward \(v_{2n-2i+3}^{2i-1} \rightarrow v_{2n-2i+3}^{2i-1}\) path \(P_1\) formed by interrow edges and inrow down edges, and the vertically downward \(v_{2n-2i+2}^{2n+2i+1} \rightarrow v_{2n-2i+3}^{2n+2i+1}\) path \(P_2\) formed by interrow edges and inrow down edges. These two paths together incur a total weight of \((n+1)B^2\), since the inrow down edges have weight 0.

• The edges \(e_i\) and \(f_i\). This incurs a weight of \(n \cdot i + n(n - i + 1) = n(n + 1)\). Finally, remove the two inrow right edges \((v_{2n-2i+2}, v_{2n-2i+2}^{2i-1})\) and \((v_{2n-2i+3}, v_{2n-2i+3}^{2i+1})\) from \(E_i\). This saves a weight of \(2B\). From the above paragraph and (4.1) it follows that the total weight of \(E_i\) is exactly \(C_w\). Note that even though we removed the edge \((v_{2n-2i+2}, v_{2n-2i+3}^{2i-1})\), we can still travel from \(v_{2n-2i+2}^{2i-1}\) to \(v_{2n-2i+2}^{2i-1}\) in \(E_i\) using the edge \(e_i\) as follows: take the path \(v_{2n-2i+2}^{2i-1} \rightarrow w_{n-1}^{n} \rightarrow v_{2n-2i+2}^{2i-1}\). Similarly, even though we removed the edge \((v_{2n-2i+3}^{2n+2i+1}, v_{2n-2i+3}^{2n+2i+3})\), we can still travel from \(v_{2n-2i+3}^{2n+2i+1}\) to \(v_{2n-2i+3}^{2n+2i+3}\) in \(E_i\) using the edge \(f_i\) as follows: take the path \(v_{2n-2i+3}^{2n+2i+1} \rightarrow w_{n-i+2}^{n} \rightarrow v_{2n-2i+3}^{2n+2i+1}\).

It remains to show that \(E_i\) satisfies the connectedness property and it represents \(i\). It is easy to see \(E_i\) represents \(i\) since the only edge in \(E_i\) which is incident to \(P\) is the edge leaving \(p_i\). Similarly, the only edge in \(E_i\) incident to \(Q\) is the one entering \(q_i\). We show that the connectedness property holds as follows (recall Definition 3.1):

1. There is a \(p_i \rightarrow p\) path in \(E_i\) by starting with the source edge leaving \(p_i\) and then following downward path \(P_1\) from \(v_{2n-2i+2}^{2i-1} \rightarrow v_{2n-2i+3}^{2i-1}\). Then travel towards the left from \(v_{2n-2i+3}^{2i-1}\) to \(p\) by using inrow left, inrow up, and inrow down edges from rows \(R_{2n-2i+2}\) and \(R_{2n-2i+3}\). Finally, use the edge \((v_{2n-2i+2}, p)\).

2. For the existence of a \(p_i \rightarrow q\) path in \(E_i\), we have seen above that there is a \(p_i \rightarrow v_{2n-2i+3}^{2i-1}\) path. Then travel towards the right from \(v_{2n-2i+3}^{2i-1}\) to \(q\) by using inrow right, inrow up, and inrow down edges from rows \(R_{2n-2i+2}\) and \(R_{2n-2i+3}\) to reach the vertex \(v_{2n-2i+2}^{2n}\). The only potential issue is that the inrow right edge \((v_{2n-2i+3}^{2n+2i+1}, v_{2n-2i+3}^{2n+2i+3})\) is missing in \(E_i\); however, this is not a problem since we have the path \(v_{2n-2i+3}^{2n+2i+1} \rightarrow w_{n-1}^{n+1} \rightarrow v_{2n-2i+3}^{2n+2i+1}\). Finally, use the edge \((v_{2n-2i+2}, q)\).

3. For the existence of a \(p \rightarrow q_i\) path in \(E_i\), first use the edge \((p, v_{2n-2i+3}^{2n+2i+3})\). Then travel towards the right by using inrow up, inrow right, and inrow down edges from rows \(R_{2n-2i+2}\) and \(R_{2n-2i+3}\) to reach the vertex \(v_{2n-2i+2}^{2n+2i+1}\). The only potential issue is that the inrow right edge \((v_{2n-2i+3}^{2n+2i+3}, v_{2n-2i+2}^{2n+2i})\) is missing in \(E_i\); however, this is not a problem since we have the path \(v_{2n-2i+3}^{2n+2i+3} \rightarrow \)
Since \( \beta \) weight of that the source and sink edges contribute at least 2 terminal edges. Suppose we have at least five terminal edges in \( E \). Let the source edge be incident to \( v_{2n-2i+2}^{2n-2i+1} \). Then travel towards the left by using inrow up, inrow left, and inrow down edges from rows \( R_{2n-2i+2} \) and \( R_{2n-2i+3} \) until you reach the vertex \( v_{2n-2i+1} \cdot 2 \). Then take the downward path \( P_2 \) from \( v_{2n-2i+2} \) to \( v_{2n-2i+1} \).

Finally, use the sink edge \((v_{2n-2i+1}^{2n-2i+1}, q_i)\) incident to \( q_i \).

4. For the existence of a \( q \to q_i \) path in \( E_i \), we first use the terminal edge \((q_i, v_{2n-2i+1}^{2n-2i+1})\). Then travel towards the left by using inrow up, inrow left, and inrow down edges from rows \( R_{2n-2i+2} \) and \( R_{2n-2i+3} \) until you reach the vertex \( v_{2n-2i+1} \). Finally, use the sink edge \((v_{2n-2i+1}^{2n-2i+1}, q_i)\) incident to \( q_i \).

Therefore, \( E_i \) indeed satisfies the connectedness property.

4.4. \( E' \) satisfies the connectedness property and has weight at most \( C_n^* \Rightarrow E' \) represents some \( \beta \in [n] \) and has weight exactly \( C_n^* \). Next we show that if a set of edges \( E' \) satisfies the connectedness property and has weight at most \( C_n^* \), then in fact the weight of \( E' \) is exactly \( C_n^* \) and it represents some \( \beta \in [n] \). We do this via the following series of claims and observations.

**Claim 4.1.** \( E' \) contains exactly one source edge and one sink edge.

**Proof.** Since \( E' \) satisfies the connectedness property it must contain at least one source edge and at least one sink edge. Without loss of generality, suppose that there are at least two source edges in \( E' \). Then the weight of \( E' \) is at least the sum of the weights of these two source edges plus the weight of at least one sink edge. Thus if \( E' \) contains at least two source edges, then its weight is at least \( B^5 \). However, from (4.1) we get that

\[
C_n^* = 2B^5 + 4B^4 + (2n + 1)B^3 + (n + 1)B^2 + (4n - 2)B + (n + 1)^2
\leq 2B^5 + 4n \cdot B^3 + 3n \cdot B^4 + 2n \cdot B^3 + 4n \cdot B^4 + 4n \cdot B^5
\leq 2B^5 + 17n \cdot B^4
< 3B^5,
\]

since \( B = 18n^2 > 17n \).

Thus we know that \( E' \) contains exactly one source edge and exactly one sink edge. Let the source edge be incident to \( p_i \) and the sink edge be incident to \( q_i \).

**Claim 4.2.** \( E' \) contains exactly four terminal edges.

**Proof.** Since \( E' \) satisfies the connectedness property, it must contain at least one incoming and one outgoing edge for both \( p \) and \( q \). Therefore, we need at least four terminal edges. Suppose we have at least five terminal edges in \( E' \). We already know that the source and sink edges contribute at least \( 2B^5 \) to the weight of \( E' \), hence the weight of \( E' \) is at least \( 2B^5 + 5B^4 \). However, from (4.1), we get that

\[
C_n^* = 2B^5 + 4B^4 + (2n + 1)B^3 + (n + 1)B^2 + (4n - 2)B + (n + 1)^2
\leq 2B^5 + 4B^4 + 3n \cdot B^3 + 2n \cdot B^3 + 4n \cdot B^4 + 4n \cdot B^5
= 2B^5 + 4B^4 + 13n \cdot B^3
< 2B^5 + 5B^4,
\]

since \( B = 18n^2 > 13n \).

Hence we know that \( E' \) contains exactly four terminal edges.

**Claim 4.3.** \( E' \) contains exactly \( 2n + 1 \) inrow up edges, one from each column \( C_2 \), for \( 0 \leq i \leq 2n \).
Proof. Observe that for each $1 \leq j \leq 2n - 1$, the inrow edges in column $C_{2j}$ form a cut between vertices from columns $C_{2j-1}$ and $C_{2j+1}$. Since $E'$ must have a path of type $p \rightsquigarrow p$ that needs to use at least one inrow up edge from each of the columns $C_0, C_2, \ldots, C_{2n-2}$. Since $E'$ must have a path of type $p \rightsquigarrow q$, path we need to use at least one inrow up edge from each of the columns $C_{2i}, C_{2i+2}, \ldots, C_{4n}$. Hence $E'$ has at least $2n + 1$ inrow up edges, as we require at least one inrow up edge from each of the columns $C_0, C_2, \ldots, C_{4n}$.

Suppose $E'$ contains at least $2n + 2$ inrow up edges. We already know that $E'$ has a contribution of $2B^5 + 4B^4$ from source, sink, and terminal edges. Hence the weight of $E'$ is at least $2B^5 + 4B^4 + (2n + 2)B^3$. However, from (4.1), we get that

$$C_n = 2B^5 + 4B^4 + (2n + 1)B^3 + (n + 1/2)B + (n + 1)^2$$

$$\leq 2B^5 + 4B^4 + (2n + 1)B^3 + 2n \cdot B^2 + 4n \cdot B^2 + 4n \cdot B^2$$

$$= 2B^5 + 4B^4 + (2n + 1)B^3 + 10n \cdot B^2$$

$$< 2B^5 + 4B^4 + (2n + 2)B^3,$$

since $B = 18n^2 > 10n$.

Therefore, we know that $E'$ contains exactly one inrow edge per column $C_i$, for every $0 \leq i \leq 2n$. By Claim 4.2, we know that exactly two terminal edges incident to $p$ are selected in $E'$. Observe that the terminal edge leaving $p$ should be followed by an inrow up edge, and similarly, the terminal edge entering $p$ follows an inrow up edge. Since we select exactly one inrow up edge from column $C_i$, it follows that the two terminal edges in $E'$ incident to $p$ must be incident to the rows $R_{2i+1}$ and $R_{2i}$, respectively for some $\ell \in [n]$. Similarly, the two terminal edges in $E'$ incident to $q$ must be incident to the rows $R_{2i' + 1}$ and $R_{2i'}$ for some $\ell' \in [n]$. We summarize this in the following claim.

**Observation 4.4.** There exist integers $\ell, \ell' \in [n]$ such that
- the only two terminal edges in $E'$ incident to $p$ are $(p, v_{2\ell+1}^0)$ and $(v_{2\ell+1}^0, p)$, and
- the only two terminal edges in $E'$ incident to $q$ are $(q, v_{2\ell'+1}^0)$ and $(v_{2\ell'+1}^0, q)$.

**Definition 4.5.** For $i \in [n + 1]$, we call the $4n$ inrow edges which connect vertices from row $R_{2i-1}$ to vertices from $R_{2i}$ as Type$(i)$ inrow edges. We can divide the Type$(i)$ inrow edges into $2n$ “pairs” of adjacent inrow edges given by $(v_{2i-1}^{2i-1}, w_i^i)$ and $(w_i^i, v_{2i}^{2i-1})$ for each $1 \leq j \leq 2n$.

Note that there are a total of $n + 1$ types of inrow edges.

**Claim 4.6.** $E'$ contains a pair of inrow edges of Type$(r)$ for each $r \in [n + 1]$. Moreover, these two edges are the only inrow edges of Type$(r)$ chosen in $E'$.

**Proof.** First we show that $E'$ contains at least one pair of inrow edges of each type. Observation 4.4 implies that we cannot avoid using inrow edges of any type by, for example, going into $p$ via an edge from some $R_{2i}$ and then exiting $p$ via an edge to some $R_{2j+1}$ for any $j > i$ (similarly for $q$). By the connectedness property, set $E'$ contains a path of type $p \rightsquigarrow p$ path $P_i$. By Observation 4.4, the only edge entering $p$ is $(v_{2i}^{2i-1}, p)$. Hence $E'$ must contain at least one pair of inrow edges of Type$(r)$ for $1 \leq r \leq \ell$ since the only way to travel from row $R_{2\ell-1}$ to $R_{2\ell}$ (for each $r \in [\ell]$) is by using a pair of inrow edges of Type$(r)$. Similarly $E'$ contains a path of type $p \rightsquigarrow q$ and the only outgoing edge from $p$ is $(p, v_{2\ell+1}^0)$. Hence $E'$ must contain at least one pair of inrow edges of Type$(r)$ for each $1 \leq r \leq n + 1$ since the only way to travel from row $R_{2\ell-1}$ to $R_{2\ell}$ is by using a pair of inrow edges of Type$(r)$. Therefore, the edge set $E'$ contains at least one pair of inrow edges of each Type$(r)$ for $1 \leq r \leq n + 1$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Next we show that \( E' \) contains exactly two interrow edges of Type(\( r \)) for each \( r \in [n + 1] \). Suppose \( E' \) contains at least three interrow edges of some Type(\( r \)) for some \( r \in [n + 1] \). Since the weight of each interrow edge is \( B^2/2 \), this implies \( E' \) gets a weight of at least \((n + 1 + \frac{1}{2}) \cdot B^2\) from the interrow edges. We have already seen \( E' \) has a contribution of \( 2B^3 + 4B^4 + (2n + 1)B^3 \) from source, sink, terminal, and inrow up edges. Hence the weight of \( E' \) is at least \( 2B^3 + 4B^4 + (2n + 1)B^3 + (n + 1 + \frac{1}{2}) \cdot B^2 \). However, from (4.1), we get that

\[
C_{n+1}^r = 2B^3 + 4B^4 + (2n + 1)B^3 + (n + 1)B^2 + (4n - 2)B + (n + 1)^2 \\
\leq 2B^3 + 4B^4 + (2n + 1)B^3 + (n + 1)B^2 + 4n \cdot B + 4n \cdot B \\
= 2B^3 + 4B^4 + (2n + 1)B^3 + (n + 1)B^2 + 8n \cdot B \\
< 2B^3 + 4B^4 + (2n + 1)B^3 + \left(n + 1 + \frac{1}{2}\right) B^2,
\]

since \( \frac{B}{2} = 9n^2 > 8n \). Hence, \( E' \) contains exactly two interrow edges of Type(\( r \)) for each \( r \in [n + 1] \).

**Claim 4.7.** For each \( r \in [n + 1] \), let the unique pair of interrow edges in \( E' \) (guaranteed by Claim 4.6) of Type(\( r \)) belong to column \( C_{2i-1} \). If the unique source and sink edges in \( E' \) (guaranteed by Claim 4.1) are incident to \( p_i \) and \( q_j' \), respectively, then we have \( i' \leq \ell_1 \leq \ell_2 \leq \cdots \leq \ell_{n+1} \leq n + j' \).

**Proof.** Observation 4.4 implies the only way to get from row \( R_{2i-1} \) to \( R_{2i} \) is to use a pair of interrow edges of Type(\( i \)). By Claim 4.6, we use exactly one pair of interrow edges of each type. Recall that there is a walk \( P = p'_i \rightsquigarrow p \rightsquigarrow q'_j \) in \( E' \), and this walk must use each of these interrow edges.

First we show that \( \ell_1 \geq i' \). Suppose \( \ell_1 < i' \leq n \). Since we use the source edge incident to \( p_i \), we must reach vertex \( v_0^{2i'-1} \) from \( v_0^{2i-1} \). Since \( i' > \ell_1 \), to use the pair of interrow edges to travel from \( v_0^{2i'-1} \) to \( v_0^{2\ell_1-1} \), the walk \( P \) must contain a \( v_0^{2i'-1} \rightsquigarrow v_0^{2\ell_1-1} \) subwalk \( P' \). By the construction of the connector gadget this subwalk \( P' \) must use the inrow up edge \( (v_0^{2\ell_1-1}, v_0^{2i'-2}) \). Now the walk \( P \) has to reach column \( C_{2n+2j'-1} \) from column \( C_{2i-1} \), and so it must use another inrow edge from column \( C_{2i'-2} \) (between rows \( R_{2i-1} \) and \( R_{2i+1} \) for some \( i \geq 1 \), which contradicts Claim 4.3.

Now we prove \( \ell_{n+1} \leq n + j' \). Suppose to the contrary that \( \ell_{n+1} > n + j' \). Then by reasoning similar to that of the above one can show that at least two inrow up edges from column \( C_{2n+2j'-2} \) are used, which contradicts Claim 4.3.

Finally suppose there exists \( r \in [n] \) such that \( \ell_r > \ell_{r+1} \). We consider the following three cases:

- \( \ell_{r+1} < \ell_r \leq n \): By using the fact that there is a \( p'_i \rightsquigarrow q'_j \) walk in \( E' \) we get at least two inrow up edges are used from column \( C_{2\ell_r-2} \), which contradicts Claim 4.3.
- \( n < \ell_r \leq n + j' \): We need to use at least two inrow up edges from column \( C_{2\ell_r-2} \), which contradicts Claim 4.3.
- \( \ell_r > n + j' \): We need to use at least two inrow up edges from column \( C_{2n+2j'-2} \), which contradicts Claim 4.3.

**Claim 4.8.** \( E' \) contains at most two shortcut edges.

**Proof.** For the proof we will use Claim 4.7. We will show that we can use at most one \( e \)-shortcut. The proof for \( f \)-shortcut is similar.

Suppose we use two \( e \)-shortcuts, viz. \( e_x \) and \( e_y \) such that \( x > y \). Note that it makes sense to include a shortcut into \( E' \) only if we use the interrow edge that
continues it. Hence \( \ell_x = x \) and \( \ell_y = y \). By Claim 4.7, we have \( y = \ell_y \geq \ell_x = x \), which is a contradiction. 

**Claim 4.9.** \( E' \) contains exactly \( 4n - 2 \) inrow right edges.

**Proof.** Since \( E' \) contains a \( p \rightarrow q \) path, it follows that \( E' \) has a path connecting some vertex from the column \( C_i \) to some vertex from column \( C_{i+1} \) for each \( 0 \leq i \leq 2n + 2j' - 2 \). Since \( E' \) contains a \( p_r \rightarrow q \) path, it follows that \( E' \) has a path connecting some vertex from the column \( C_j \) to some vertex from the column \( C_{j+1} \) for each \( 2i' - 1 \leq j \leq 4n - 1 \).

Since \( 2n + 2j' - 2 \geq 2n \) and \( 2i' - 1 \leq 2n \), it follows that for each \( 0 \leq i \leq 4n - 1 \) the solution \( E' \) must contain a path connecting some vertex from column \( C_i \) to some vertex from column \( C_{i+1} \). Each such path has to either be a path of one which must be an inrow right edge, or a path of two edges consisting of a shortcut and an interrow edge. But Claim 4.8 implies \( E' \) contains at most two shortcuts. Therefore, \( E' \) contains at least \( 4n - 2 \) inrow right edges. Suppose \( E' \) contains at least \( 4n - 1 \) inrow right edges. We have already seen the contribution of source, sink, terminal, inrow up, and interrow edges is \( 2B^3 + 4B^4 + (2n+1)B^3 + (n+1)B^2 \). If \( E' \) contains at least \( 4n - 1 \) inrow right edges, then the weight of \( E' \) is at least \( 2B^3 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-1)B \).

However, from \((4.1)\), we get that 

\[
C_n^* = 2B^3 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + (n+1)^2
\]

\[
= 2B^3 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-2)B + 4n^2
\]

\[
< 2B^3 + 4B^4 + (2n+1)B^3 + (n+1)B^2 + (4n-1)B,
\]

since \( B = 18n^2 > 4n^2 \).

From Claim 4.8 and the proof of Claim 4.9, we can make the following observation.

**Observation 4.10.** \( E' \) contains exactly two shortcuts.

Let the shortcuts used be \( e_{i'} \) and \( f_{j'} \). Recall that Claim 4.1 implies that at most one edge incident to \( P \) and at most one edge incident to \( Q \) is used in \( E' \). Therefore, if we show that \( i' = j' \), then it follows that \( E' \) represents \( \beta = i' = j' \).

**Claim 4.11.** The following inequalities hold:

- \( i'' \geq i' \) and \( j'' \leq j' \);
- \( i'' \geq j'' \).

**Proof.** To use the shortcut \( e_{i'} \), we need to use the lower half of a pair of interrow edges from column \( C_{2n+2j' - 1} \). Claim 4.7 implies \( i' \leq \ell_1 \) and the pairs of interrow edges used are monotone from left to right. Hence \( i'' \geq i' \). Similarly, to use the shortcut \( f_{j'} \), we need to use the upper half of an interrow edge from Column \( C_{2n+2j'-1} \). Claim 4.7 implies \( n + j' \geq \ell_{n+1} \geq n + j'' \). Hence \( j'' \leq j' \).

Since we use the shortcut \( e_{i'} \) it follows that \( \ell_{n-i''+1} = i'' \). Similarly, since we use the shortcut \( f_{j''} \) it follows that \( \ell_{n-j''+2} = n+j'' \). As \( 1 \leq i'', j'' \leq n \) it follows that \( n + j'' > i'' \). By monotonicity of the \( \ell \)-sequence shown in Claim 4.7, we have \( n - j'' + 2 > n - i'' + 1 \), i.e., \( i'' \geq j'' \).

**Theorem 4.12.** The weight of \( E' \) is exactly \( C_n^* \), and \( E' \) represents some integer \( \beta \in [n] \).

**Proof.** As argued above it is enough to show that \( i' = j' \). We have already seen \( E' \) has the following contribution to its weight:
The source edge incident to $p_{i'}$ has weight $B^5 + (n - i' + 1)$ by Claim 4.1.
The sink edge incident to $q_{j''}$ has weight $B^5 + j''$ by Claim 4.1.
The terminal edges incur weight $4B^4$ by Claim 4.2.
The inrow up edges incur weight $(2n + 1)B^3$ by Claim 4.3.
The interrow edges incur weight $(n + 1)B^2$ by Claim 4.6.
The inrow right edges incur weight $(4n - 2)B$ by Claim 4.9.
The shortcut $e_{ii'}$ incurs weight $n \cdot i''$ and $f_{jj''}$ incurs weight $n(n - j'' + 1)$ by Observation 4.10.

Thus we already have a weight of (4.2)

$$C^{**} = (2B^5 + (n - i' + j' + 1)) + 4B^4 + (2n + 1)B^3 + (n + 1)B^2 + (4n - 2)B + n(n - j'' + i'' + 1).$$

Observe that adding any edge of nonzero weight to $E'$ (other than the ones mentioned above) increases the weight $C^{**}$ by at least $B$, since Claim 4.8 does not allow us to use any more shortcuts. Equations (4.1) and (4.2) imply $C^{**} + B - C_n^* = B - n(i' - j' - (j'' - i'')) \geq 20n^3 - n(i' - j') - (j'' - i'') \geq 0$, since $i', i'', j', j'' \in [n]$. This implies that the weight of $E'$ is exactly $C^{**}$. We now show that in fact $C^{**} - C_n^* \geq 0$, which will imply that $C^{**} = C_n^*$. From (4.1) and (4.2), we have $C^{**} - C_n^* = (j' - i') + n(i'' - j'')$.

We now show that this quantity is nonnegative. Recall that from Claim 4.11, we have $i'' \geq j''$.

- If $i'' > j''$ then $n(i'' - j'') \geq n$. Since $j', i' \in [n]$, we have $j' - i' \geq 1 - n$.
  Therefore, $(j' - i') + n(i'' - j'') \geq n + (1 - n) = 1$.
- If $i'' = j''$ then by Claim 4.11 we have $i' \leq i'' = j''$. Hence $(j' - i') \geq 0$ and so $(j' - i') + n(i'' - j'') \geq 0$.

Therefore $C^{**} = C_n^*$, i.e., $E'$ has weight exactly $C_n^*$. However $C_n^* = C^{**}$ implies

$$j' - i' + n(i'' - j'') = 0.$$

Since $i', j' \in [n]$ we have $n - 1 \geq j' - i' \geq 1 - n$. If $i'' \neq j''$ then $n(i'' - j'') \geq n$ and hence $j' - i' + n(i'' - j'') \geq (1 - n) + n \geq 1$. This is a contradiction. Hence, we have $j'' = i''$ and therefore (4.3) implies $j' = i'$, i.e., $E'$ is represented by $i' = j' \in [n]$.

5. Proof of Lemma 3.6: Constructing the main gadget. We prove Lemma 3.6 in this section, by constructing a main gadget satisfying the specifications of section 3.2. Recall that, as discussed at the start of section 3.3, we may assume that $1 < x, y < n$ holds for every $(x, y) \in S_{i,j}$.

5.1. Different types of edges in main gadget. Before proving Lemma 3.6, we first describe the construction of the main gadget in more detail (see Figure 4). The main gadget has $n^2$ rows denoted by $R_1, R_2, \ldots, R_{n^2}$ and $2n + 1$ columns denoted by $C_0, C_1, \ldots, C_{2n+1}$. Let us denote the vertex at the intersection of row $R_i$ and column $C_j$ by $v_{ij}$. We now describe the various different kinds of edges in the main gadget.

1. **Left source edges**: For every $i \in [n]$, the edge $(\ell_i, \ell'_i)$ is a left source edge.
2. **Right sink edges**: For every $i \in [n]$, the edge $(r'_i, r_i)$ is a right sink edge.
3. **Top source edges**: For every $i \in [n]$, the edge $(t_i, v'_i)$ is a top source edge.
4. **Bottom sink edges**: For every $i \in [n]$, the edge $(s_{n+1}, b_i)$ is a bottom sink edge.
5. **Source internal edges**: This is the set of $n^2$ edges of the form $(\ell'_i, v'_i)$ for $i \in [n]$ and $n(i - 1) + 1 \leq j \leq n \cdot i$. We number the source internal edges from top to bottom, i.e., the edge $(\ell'_i, v'_i)$ is called the $j$th source internal edge, where $i \in [n]$ and $n(i - 1) + 1 \leq j \leq n \cdot i$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Fig. 4. The main gadget (for $n = 4$) representing the set $\{(2, 2), (2, 3), (3, 2)\}$. The highlighted edges represent the pair $(2, 3)$. 
6. Sink internal edges: This is the set of \( n^2 \) edges of the form \((v_i^{2n+1}, r_i')\) for 
\( i \in [n] \) and \( n(i-1) + 1 \leq j \leq n - i \). We number the sink internal edges from 
top to bottom, i.e., \( (v_j^{2n+1}, r_i') \) is called \( j \)th sink internal edge, 
where \( i \in [n] \) and \( n(i-1) + 1 \leq j \leq n - i \).

7. Bridge edges: This is the set of \( n^2 \) edges of the form \((v_i^n, v_i^{n+1})\) for 
\( 1 \leq i \leq n^2 \). We number the bridge edges from top to bottom, i.e., \( (v_i^n, v_i^{n+1}) \) is 
called \( i \)th bridge edge. These edges are shown in red in Figure 4.

8. Inrow right edges: For each \( i \in [n^2] \) we call the \( \rightarrow \) edges (except the 
bridge edge \((v_i^n, v_i^{n+1})\)) connecting vertices of row \( R_i \) \( \rightarrow \) inrow right edges. Formally, 
the set of inrow right edges of row \( R_i \) are given by \( \{(v_i^1, v_i^{j+1}) : 0 \leq j \leq n-1\} \cup \{(v_i^j, v_i^{j+1}) : n+1 \leq j \leq 2n\} \).

9. Interrow down edges: For each \( i \in [n^2] \) we call the \( 2n \downarrow \) edges connecting 
vertices of row \( R_i \) \( \rightarrow \) interrow down edges. The \( 2n \) interrow edges 
between rows \( R_i \) \( \rightarrow \) and \( R_{i+1} \) are \((v_i^j, v_{i+1}^j)\) for each \( 1 \leq j \leq 2n \).

10. Shortcut edges: There are \( 2|S| \) shortcut edges, namely, \( e_1, e_2, \ldots, e_{|S|} \) and 
\( f_1, f_2, \ldots, f_{|S|} \). The shortcut edge for an \((x, y) \in S\) for some \( 1 \leq x, y < n \) is 
defined in the following way:

- Introduce a new vertex \( g_{x,y}^y \) at the middle of the edge \((r_{n(x-1)+y}, v_{n(x-1)+y})\) to create two new edges \((v_{n(x-1)+y}, g_{x,y}^y)\) and \((g_{x,y}^y, v_{n(x-1)+y+1})\).

- Introduce a new vertex \( h_{x,y}^y \) at the middle of the edge \((v_{n(x-1)+y}, v_{n(x-1)+y+1})\) to create two new edges \((v_{n(x-1)+y}, h_{x,y}^y)\) and \((h_{x,y}^y, v_{n(x-1)+y+1})\).

5.2. Assigning weights in the main gadget. Define \( B = 11n^2 \). We assign 
weights to the edges as follows:

1. Each left source edge has weight \( B^4 \).
2. Each right sink edge has weight \( B^4 \).
3. For every \( 1 \leq i \leq n \), the \( i \)th top source edge \((t_i, v_i^1)\) has weight \( B^4 \).
4. For every \( 1 \leq i \leq n \), the \( i \)th bottom sink edge \((v_i^{n+1}, b_i)\) has weight \( B^4 \).
5. For each \( i \in [n^2] \), the \( i \)th bridge edge \((v_i^n, v_i^{n+1})\) has weight \( B^3 \).
6. For each \( i \in [n^2] \), the \( i \)th source internal edge has weight \( B^3(n^2 - i) \).
7. For each \( j \in [n^2] \), the \( j \)th sink internal edge has weight \( B^3 \).
8. Each inrow right edge has weight \( 3B \).
9. For each \((x, y) \in S\), both the shortcut edges \( e_{x,y} \) and \( f_{x,y} \) have weight \( B \).
10. Each interrow down edge that does not have a shortcut incident to it has weight \( 2 \).

If an interrow edge is split into two edges by the shortcut incident to it, then we assign a weight 1 to each of the two parts.

Now we define the quantity \( M_n^* \) stated in Lemma 3.6:

\[
M_n^* = 4B^4 + B^3 + B^2n^2 + B(6n - 4) + 2(n^2 - 1).
\]

Next we are ready to prove the statements of Lemma 3.6.

5.3. For every \((x, y) \in S\), there is a solution \( E_{x,y} \) of weight \( M_n^* \) that 
represents \((x, y)\). For \((x, y) \in S \subseteq [n] \times [n] \) define \( z = n(x-1) + y \). Let \( E_{x,y} \) be 
the union of the following sets of edges:

- The \( 2 \)th left source edge and \( 2 \)th right sink edge. This incurs a weight of 
\( 2B^4 \).
• The \( g \)th top source edge and the \( g \)th bottom sink edge. This incurs a weight of \( 2B^4 \).
• The \( h \)th bridge edge. This incurs a weight of \( B^3 \).
• The \( i \)th source internal edge and \( j \)th sink internal edge. This incurs a weight of \( B^2n^3 \).
• All inrow right edges from row \( R_z \) except \( (v_z^{y-1}, v_z^y) \) and \( (v_z^{n+y}, v_z^{n+y+1}) \). This incurs a weight of \( 3B \cdot (2n - 2) \).
• The shortcut edges \( e_{x,y} \) and \( f_{x,y} \). This incurs a weight of \( 2B \).
• The vertically downward path \( v_z^y \rightarrow v_z^{y+1} \rightarrow \cdots \rightarrow v_z^{n+y} \). This incurs a weight of \( 2(n - 1) \).
• The vertically downward path \( v_z^{n+y} \rightarrow v_z^{n+y+1} \rightarrow \cdots \rightarrow v_z^{n+y+1} \). This incurs a weight of \( 2(n^2 - z) \).

From the above paragraph and (5.1), it follows the total weight of \( E_{x,y} \) is exactly \( M_{n^3} \). Note that we did not include two inrow right edges, \( (v_z^{y-1}, v_z^y) \) and \( (v_z^{n+y}, v_z^{n+y+1}) \), from row \( R_z \) in \( E_{x,y} \). However, we can mimic the function of both these inrow right edges using shortcut edges and interrow down edges in \( E_{x,y} \) as follows:

- We can still travel from \( v_z^y \) to \( v_z^{y+1} \) in \( E_{x,y} \) as follows: take the path \( (v_z^{y-1} \rightarrow g_z^y \rightarrow v_z^y) \).
- We can still travel from \( v_z^{n+y} \) to \( v_z^{n+y+1} \) in \( E_{x,y} \) via the path \( (v_z^{n+y} \rightarrow h_z \rightarrow v_z^{n+y+1}) \).

The following observation follows from the previous paragraph.

Observation 5.1. In \( E_{x,y} \) we can reach \( v_z^j \) from \( v_z^i \) for any \( 2n + 1 \geq j \geq i \geq 0 \).

It remains to show that \( E_{x,y} \) represents \( (x, y) \in S \). It is easy to see that the first four conditions of Definition 3.5 are satisfied since the definition of \( E_{x,y} \) itself gives the following:

- In \( E_{x,y} \) the only outgoing edge from \( L \) is the one incident to \( \ell_x \).
- In \( E_{x,y} \) the only incoming edge to \( R \) is the one incident to \( r_y \).
- In \( E_{x,y} \) the only outgoing edge from \( T \) is the one incident to \( t_y \).
- In \( E_{x,y} \) the only incoming edge to \( B \) is the one incident to \( b_x \).

We now show that the last condition of Definition 3.5 is also satisfied by \( E_{x,y} \):

1. There is an \( \ell_x \equiv r_z \) path in \( E_{x,y} \) obtained by taking the edges in the following order:
   - the left source edge \( (\ell_x, \ell_x') \);
   - the source internal edge \( (\ell_x', v_x^0) \);
   - the horizontal path \( v_x^0 \rightarrow v_x^1 \rightarrow \cdots \rightarrow v_x^n \) given by Observation 5.1;
   - the bridge edge \( (v_x^n, v_x^{n+1}) \);
   - the horizontal path \( v_x^{n+1} \rightarrow v_x^{n+2} \rightarrow \cdots \rightarrow v_x^{2n+1} \) given by Observation 5.1;
   - the sink internal edge \( (v_x^{2n+1}, \ell_x) \);
   - the right sink edge \( (r_x', r_x) \).

2. There is a \( t_y \equiv b_y \) path in \( E_{x,y} \) obtained by taking the edges in the following order:
   - the top source edge \( (t_y, v_y^1) \);
   - the downward path \( v_y^1 \rightarrow v_y^2 \rightarrow \cdots \rightarrow v_y^n \) given by interrow down edges in column \( C_y \);
   - the horizontal path \( v_y^n \rightarrow v_y^{n+1} \rightarrow \cdots \rightarrow v_y^{n} \) given by Observation 5.1;
   - the bridge edge \( (v_y^n, v_y^{n+1}) \);
   - the horizontal path \( v_y^{n+1} \rightarrow v_y^{n+2} \rightarrow \cdots \rightarrow v_y^{n+y} \) given by Observation 5.1;

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
the downward path \( v_z^{n+y} \rightarrow v_{z+1}^{n+y} \rightarrow \cdots v_{n+1}^{n+y} \) given by interrow down edges in column \( C_{n+y} \) and

- the bottom sink edge \( (v_{n+1}^{n+y}, b_y) \).

Therefore, \( E_{x,y} \) has weight \( M_n^\prime \) and represents \( (x, y) \).

5.4. \( E' \) satisfies the connectedness property and has weight at most \( M_n^\ast \Rightarrow E' \) represents some \((\alpha, \beta) \in S\) and has weight exactly \( M_n^\ast \). In this section we show that if a set of edges \( E' \) satisfies the connectedness property and has weight \( M_n^\ast \), then it represents some \((\alpha, \beta) \in S\). We do this via the following series of claims and observations.

**Claim 5.2.** \( E' \) contains

- exactly one left source edge,
- exactly one right sink edge,
- exactly one top source edge, and
- exactly one bottom sink edge.

**Proof.** Since \( E' \) satisfies the connectedness property, it must contain at least one edge of each of the above types. Without loss of generality, suppose we have at least two left source edges in \( E' \). Then the weight of the edge set \( E' \) is greater than or equal to the sum of weights of these two left source edges and the weight of a right sink edge, the weight of a top source edge, and the weight of a bottom sink edge. Thus if \( E' \) contains at least two left source edges, then its weight is at least \( 5B^4 \). However, from (5.1), we get that

\[
M_n^\ast = 4B^4 + B^3 + B^2n^2 + B(6n - 4) + 2(n^2 - 1) \\
\leq 4B^4 + n \cdot B^3 + n \cdot B^3 + 6n \cdot B^3 + 2n \cdot B^3 \\
= 4B^4 + 10n \cdot B^3 \\
< 5B^4,
\]

since \( B = 11n^2 > 10n \).

Therefore, we can set up the following notation:

- Let \( i_L \in [n] \) be the unique index such that the left source edge in \( E' \) is incident to \( \ell_{i_L} \).
- Let \( i_R \in [n] \) be the unique index such that the right sink edge in \( E' \) is incident to \( r_{i_R} \).
- Let \( i_T \in [n] \) be the unique index such that the top source edge in \( E' \) is incident to \( \ell_{i_T} \).
- Let \( i_B \in [n] \) be the unique index such that the bottom sink edge in \( E' \) is incident to \( b_{i_B} \).

**Claim 5.3.** The edge set \( E' \) contains exactly one bridge edge.

**Proof.** To satisfy the connectedness property, we need at least one bridge edge, since the bridge edges form a cut between the top-distinguished vertices and the right-distinguished vertices as well as between the top-distinguished vertices and the bottom-distinguished vertices. Suppose that the edge set \( E' \) contains at least two bridge edges. This contributes a weight of \( 2B^3 \). We already have a contribution on \( 4B^4 \) to the weight of \( E' \) from Claim 5.2. Therefore, the weight of \( E' \) is at least
$4B^4 + 2B^3$. However, from (5.1), we get that

\[ M_n^* = 4B^4 + B^3 + B^2n^2 + B(6n - 4) + 2(n^2 - 1) \]
\[ \leq 4B^4 + B^3 + B^2n^2 + 6n \cdot B + 2n^2 \]
\[ \leq 4B^4 + B^3 + B^2n^2 + 6n^2B^2 + 2n^2B^2 \]
\[ = 4B^4 + B^3 + 9B^2n^2 \]
\[ < 4B^8 + 2B^3, \]

since $B = 11n^2 > 9n^2$. \hfill \Box

Let the index of the unique bridge edge in $E'$ (guaranteed by Claim 5.3) be $\gamma \in [n^2]$. The connectedness property implies that we need to select at least one source internal edge incident to $v'_{in}$ and at least one sink internal edge incident to $r'_{in}$. Let the index of the source internal edge incident to $v'_{in}$ be $j_L$ and the index of the sink internal edge incident to $r'_{in}$ be $j_R$.

**Claim 5.4.** $i_L = i_R$ and $j_L = j_R = \gamma$.

**Proof.** By the connectedness property, there is a path from $v'_{in}$ to some vertex in $r_{in} \cup b_{in}$. The path starts with $v'_{in} \rightarrow v'_{in} \rightarrow v'_{in}$ and has to use the $\gamma$th bridge edge. By the construction of the main gadget (all edges are either downwards or towards the right), this path can never reach any row $R_{\ell}$ for $\ell < j_L$. Therefore, $\gamma \geq j_L$. By similar logic, we get $j_R \geq \gamma$. Therefore $j_R \geq j_L$.

If $j_R > j_L$, then the weight of the source internal edge and the sink internal edge is $B^3(n^2 - j_L + j_R) \geq B^3(n^2 + 1)$. We already have a contribution of $4B^4 + B^3$ to the weight of $E'$ from Claims 5.2 and 5.3. Therefore, the weight of $E'$ is at least $4B^4 + B^3 + B^2(n^2 + 1)$. However, from (5.1), we get that

\[ M_n^* = 4B^4 + B^3 + B^2n^2 + B(6n - 4) + 2(n^2 - 1) \]
\[ \leq 4B^4 + B^3 + B^2n^2 + 6n \cdot B + 2n^2 \]
\[ \leq 4B^4 + B^3 + B^2n^2 + 6n^2 \cdot B + 2n^2 \cdot B \]
\[ = 4B^4 + B^3 + B^2n^2 + 8n^2 \cdot B \]
\[ < 4B^8 + 2B^3, \]

since $B = 11n^2 > 8n^2$. Hence $j_R = j_L = \gamma$. Observing that $i_L = \lceil \frac{11n^2}{B} \rceil$ and $i_R = \lceil \frac{11n^2}{B} \rceil$, we obtain $i_L = i_R$. \hfill \Box

Let $i_L = i_R = \alpha$ and $\gamma = n(\alpha - 1) + \beta$. We will now show that $E'$ represents the pair $(\alpha, \beta)$. By Definition 3.5, we need to prove the following four conditions:

1. The only left source edge in $E'$ is the one incident to $t_{\alpha}$ and the only right sink edge in $E'$ is the one incident to $r_{\alpha}$.
2. The pair $(\alpha, \beta)$ is in $S$.
3. The only top source edge in $E'$ is the one incident to $t_{\beta}$ and the only bottom sink edge in $E'$ is the one incident to $b_{\beta}$.
4. $E'$ has an $t_{\alpha} \sim r_{\alpha}$ path and an $t_{\beta} \sim b_{\beta}$ path.

The first statement above follows from Claims 5.2 and 5.4. We now continue with the proof of the other three statements mentioned above.
As a corollary, we get that there are two shortcuts incident to row \( R_\gamma \). A corollary of Claim 5.5, all inrow horizontal edges are only from row \( R_\gamma \) and column \( C_\beta \) to replace an inrow right edge. Hence \( P_1 \) uses at least \( n - 1 \) inrow right edges, with equality only if \( R_\gamma \) has a shortcut incident to it.

Similarly, the \( \ell_\alpha \) \( \rightsquigarrow \) \( r_\alpha \) path in \( E' \) contains a \( v^{n+1}_\alpha \) \( \rightsquigarrow \) \( v^n_{\alpha +1} \) subpath \( P_2 \). By the construction of the main gadget, we cannot reach an upper row from a lower row. Hence this subpath \( P_2 \) must be the path \( v^{n+1}_\alpha \) \( \rightsquigarrow \) \( v^{n+2}_\alpha \) \( \rightsquigarrow \) \( \cdots \) \( \rightsquigarrow \) \( v^{2n+1}_\alpha \). This path \( P_2 \) can at most use the unique shortcut edge incident to row \( R_\gamma \) and column \( C_\beta \) to replace an inrow right edge. Hence \( P_2 \) uses at least \( n - 1 \) inrow right edges, with equality only if \( R_\gamma \) has a shortcut incident to it.

Clearly, the sets of inrow edges used by \( P_1 \) and \( P_2 \) are disjoint and, hence, \( E' \) contains at least \( 2n - 2 \) inrow right edges from row \( R_\gamma \). Suppose \( E' \) contains at least \( 2n - 1 \) inrow right edges. Then it incurs a weight of \( 3B \cdot (2n - 1) \). From Claims 5.2, 5.3, and 5.4 we already have a contribution of \( 4B^4 + B^3 + B^2n^2 \). Therefore the weight of \( E' \) is at least \( 4B^4 + B^3 + B^2n^2 + 3B \cdot (2n - 1) \).

However, from (5.1), we get that

\[
M^*_n = 4B^4 + B^3 + B^2n^2 + B(6n - 4) + 2(n^2 - 1)
\]

\[
\leq 4B^4 + B^3 + B^2n^2 + B(6n - 4) + 2n^2
\]

\[
< 4B^4 + B^3 + B^2n^2 + 3B \cdot (2n - 1),
\]

since \( B = 11n^2 > 2n^2 \). Therefore, \( E' \) can only contain at most \( 2n - 2 \) inrow right edges. Hence there must be two shortcut edges incident to row \( R_\gamma \), which are both used by \( E' \). Since \( \gamma = n(\alpha - 1) + \beta \), the fact that row \( R_\gamma \) has shortcut edges incident to it implies \( (\alpha, \beta) \in S \).

To prove the third claim it is sufficient to show that \( i_T = i_B = \beta \), since Claim 5.2 implies \( E' \) contains exactly one top source edge and exactly one bottom sink edge. Note that the remaining budget left for the weight of \( E' \) is at most \( 2(n^2 - 1) \).

**Claim 5.6.** \( i_T = i_B = \beta \).

**Proof.** Recall that the only bridge edge used is the one on row \( R_\gamma \). Moreover, the bridge edges form a cut between \( T \) and \( R \cup B \). Hence, to satisfy the connectedness property it follows that the \( b_\gamma \) \( \rightsquigarrow \) \( r_\alpha \cup b_\alpha \) path in \( E' \) contains a \( v^{2r}_1 \) \( \rightsquigarrow \) \( v^r_1 \) subpath \( P_3 \). By Claim 5.5, all inrow right edges are only from row \( R_\gamma \). As the only remaining budget is \( 2(n^2 - 1) \), we cannot use any other shortcuts or inrow right edges since \( B = 11n^2 > 2(n^2 - 1) \). Therefore, \( P_3 \) contains another \( v^{r}_1 \) \( \rightsquigarrow \) \( v^{r}_1 \) subpath \( P'_3 \). If \( i_T \neq \beta \), then \( P'_3 \) incurs weight \( 2(\gamma - 1) \). Note that we also pay a weight of 1 to use half of the interrow edge when we use the shortcut edge (which we have to use due to Claim 5.5) which is incident to row \( R_\gamma \) and column \( C_\beta \).

Similarly, the \( \ell_\alpha \) \( \cup \) \( \ell_\alpha \) \( \rightsquigarrow \) \( b_\alpha \) path in \( E' \) contains a \( v^{n+1}_\alpha \) \( \rightsquigarrow \) \( v^{n+1}_\alpha \) subpath \( P'_4 \). By Claim 5.5, all inrow horizontal edges are only from row \( R_\gamma \). As the only remaining budget is \( 2(n^2 - 1) \), we cannot use any other shortcuts or inrow right edges. Therefore,
$P_i$ contains another $v_i^{n+1} \rightarrow v_i^2$ subpath $P'_i$. If $i_B \neq \beta$, then $P'_i$ incurs weight 2$(n^2 - \gamma)$. Note that we also pay a weight of 1 to use (half of) the interrow edge when we use the shortcut edge (which we have to use due to Claim 5.5) which is incident to row $R_i$ and column $C_{n+\beta}$.

Suppose without loss of generality that $i_T \neq \beta$. Then $P'_i$ incurs a weight of 2$(\gamma - 1)$, and the half interrow edge used incurs an additional weight of 1. In addition, path $P'_i$ incurs a weight of 2$(n^2 - \gamma)$. Hence the total weight incurred is 2$(\gamma - 1) + 1 + 2(n^2 - \gamma) = 2(n^2 - 1) + 1$ which is greater than our allowed budget. Hence $i_T = \beta$. It can be shown similarly that $i_B = \beta$. \hfill \Box

**Claim 5.7.** $E'$ has an $\ell_\alpha \rightsquigarrow r_\alpha$ path and a $t_\beta \rightsquigarrow b_\beta$ path.

**Proof.** First we show that $E'$ has an $\ell_\alpha \rightarrow r_\alpha$ path by taking the following edges (in order):
- The path $\ell_\alpha \rightarrow \ell_\alpha' \rightarrow v_0^\alpha$ which exists since $i_L = \alpha$ and $j_L = \gamma$.
- The $v_0^\alpha \leftrightarrow v_0^\alpha$ path $P_1$ guaranteed in the proof of Claim 5.5.
- The bridge edge $v_0^\alpha \rightarrow v_0^{\gamma+1}$ guaranteed by Claim 5.3.
- The $v_0^{\gamma+1} \rightarrow v_0^{n+1}$ path $P_2$ guaranteed in the proof of Claim 5.5.
- The path $r_\alpha \rightarrow r_\alpha' \leftrightarrow v_0^{n+1}$ which exists since $i_R = \alpha$ and $j_R = \gamma$.

Next we show that $E'$ has a $t_\beta \rightsquigarrow b_\beta$ path by taking the following edges (in order):
- The edge $t_\beta \rightarrow v_1^\gamma$ which exists since $i_T = \beta$.
- The $v_1^\gamma \leftrightarrow v_1^\gamma$ path $P_3$ guaranteed in the proof of Claim 5.6.
- The bridge edge $v_1^\gamma \rightarrow v_1^{\gamma+1}$ guaranteed by Claim 5.3.
- The $v_1^{\gamma+1} \rightarrow v_1^{n+1}$ path $P_4$ guaranteed in the proof of Claim 5.6.
- The edge $b_\beta \rightarrow v_1^{n+1}$ which exists since $i_B = \beta$. \hfill \Box

Claims 5.2, 5.4, 5.5, 5.6, and 5.7 together imply that $E'$ represents $(\alpha, \beta) \in S$ (see Definition 3.5). We now show that the weight of $E'$ is exactly $M_n^*$.

**Lemma 5.8.** Weight of $E'$ is exactly $M_n^*$

**Proof.** Claim 5.2 contributes a weight of $4B^4$ to $E'$. Claim 5.3 contributes a weight of $B^3$ to $E'$. From the proof of Claim 5.4, we can see that $E'$ incurs weight $B^3n^2$ from the source internal edge and sink internal edge. Claim 5.5 implies that $E'$ contains exactly $2n - 2$ inrow right edges from row $R_s$ and also both shortcuts incident to row $R_s$. This incurs a cost of $3B(2n-2)+2B = B(6n-4)$. By arguments similar to that in the proof of Claim 5.6, $E'$ contains at least $(\gamma - 1)$ interrow edges from column $C_3$ and at least $(n^2 - \gamma)$ interrow edges from column $C_{n+\beta}$. Therefore, we have a weight of $E' \geq 4B^4 + B^3 + B^2n^2 + B \cdot (6n - 4) + 2(\gamma - 1) + 2(n^2 - \gamma) = 4B^4 + B^3 + B^2n^2 + B \cdot (6n - 4) + 2(n^2 - 1) = M_n^*$. Hence the weight of $E'$ is exactly $M_n^*$.

This completes the proof of the second statement of Lemma 3.6.

### 6. W[1]-hardness for SCSS in general graphs.

The main goal of this section is to prove Theorem 1.3. We note that the reduction of Guo, Niedermeier, and Suchý [42] gives a reduction from MULTICOLORED CLIQUE which builds an equivalent instance of SCSS with quadratic blowup in the number of terminals. Hence using the reduction of Guo, Niedermeier, and Suchý [42] only an $f(k) \cdot n^{o(\sqrt{k})}$ algorithm for SCSS can be ruled out under ETH. We are able to improve upon this hardness by using the PSI problem introduced by Marx [57]. Our reduction is also slightly simpler than the one given by Guo, Niedermeier, and Suchý.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
The PSI problem is so-called because the vertices of $H$ are partitioned into parts: one part corresponding to every vertex of $G$. Marx [57] showed the following hardness result.

**Theorem 6.1.** Unless ETH fails, PSI cannot be solved in time $f(r) \cdot n^{o(r/\log r)}$, where $f$ is any computable function, $r$ is the number of edges in $G$, and $n$ is the number of vertices in $H$.

By giving a reduction from PSI to SCSS, where $k = O(|E_G|)$, we will obtain an $f(k) \cdot n^{o(k/\log k)}$ hardness for SCSS under the ETH, where $k$ is the number of terminals. Consider an instance $(G, H)$ of PSI. We now build an instance $(G^*, T^*)$ of SCSS as follows:

- $B = \{b_i \mid i \in [\ell]\}$.
- $C = \{c_v \mid v \in V_H\}$.
- $H = \{h_v \mid v \in V_H\}$.
- $D = \{d_{uv} \cup d_{vu} \mid \{u, v\} \in E_H\}$.
- $A = \{a_{uv} \cup a_{vu} \mid \{u, v\} \in E_H\}$.
- $F = \{f_{ij} \mid 1 \leq i, j \leq \ell \mid g_i g_j \in E_G\}$.
- $V^* = B \cup C \cup H \cup D \cup A \cup F$.
- $E_1 = \{(c_v, b_i) \mid v \in H_i, 1 \leq i \leq \ell\}$.
- $E_2 = \{(b_i, h_v) \mid v \in H_i, 1 \leq i \leq \ell\}$.
- $E_3 = \{(h_v, c_v) \mid v \in V_H\}$.
- $E_4 = \{(c_v, d_{uv}) \mid \{u, v\} \in E_H\}$.
- $E_5 = \{(a_{uv}, h_u) \mid \{u, v\} \in E_H\}$.
- $E_6 = \{(d_{uv}, a_{vu}) \mid \{u, v\} \in E_H\}$.
- $E_7 = \{(f_{ij}, d_{uv}) \cup (a_{vu}, f_{ij}) \mid \{u, v\} \in E_H; v \in H_i; u \in H_j; 1 \leq i, j \leq \ell\}$.
- $E^* = E_1 \cup E_2 \cup E_3 \cup E_4 \cup E_5 \cup E_6 \cup E_7$.
- The set of terminals is $T^* = B \cup F$.

This completes the construction of the graph $G^* = (V^*, E^*)$. An illustration of the construction for a small graph is given in Figure 5. In the instance of PSI we can assume the graph $G$ is connected, otherwise we can solve the problem for each connected component. Therefore, we have that $k = |T| = \ell + 2|E_G| = O(|E_G|)$. For ease of argument, we distinguish the different types of edges of $G^*$ as follows (see Figure 5):

- Edges of $E_1 \cup E_2 \cup E_3$ are denoted using black edges.
- Edges of $E_4 \cup E_5$ are denoted using light/gray edges.
- Edges of $E_6 \cup E_7$ are denoted using dotted edges.

We now show two lemmas which complete the reduction from PSI to SCSS.

**Lemma 6.2.** If the instance $(G, H)$ of PSI answers YES then the instance $(G^*, T^*)$ of SCSS has a solution of size $\leq 3\ell + 10|E_G|$.

---

Partitioned Subgraph Isomorphism (PSI)

**Input:** Undirected graphs $G = (V_G = \{g_1, g_2, \ldots, g_\ell\}, E_G)$ and $H = (V_H, E_H)$, and a partition of $V_H$ into disjoint subsets $H_1, H_2, \ldots, H_\ell$

**Question:** Is there an injection $\phi : V_G \to V_H$ such that

1. for every $i \in [\ell]$ we have $\phi(g_i) \in H_i$;
2. for every edge $(g_i, g_j) \in E_G$ we have $(\phi(g_i), \phi(g_j)) \in E_H$.
Proof. Suppose the instance \((G, H)\) of PSI answers YES and let \(\phi\) be the injection from \(V_G \rightarrow V_H\). Then we claim the following set \(M'\) of \(3\ell + 10\lvert E_G\rvert\) edges forms a solution for the SCSS instance:

- \(M_1 = \{(h_{\phi(g)}, c_{\phi(g)}) \mid i \in [\ell]\}\).
- \(M_2 = \{(h_i, h_{\phi(g)}) \mid i \in [\ell]\}\).
- \(M_3 = \{(c_{\phi(g)}, b_i) \mid i \in [\ell]\}\).
- \(M_4 = \{(c_{\phi(g)}, d_{\phi(g)}) \cup (d_{\phi(g)}, a_{\phi(g)}) \cup (a_{\phi(g)}, h_{\phi(g)}) \mid g, g_j \in E_G; 1 \leq i, j \leq \ell\}\).
- \(M_5 = \{(f_{ij}, d_{\phi(g)}) \cup (a_{\phi(g)}, f_{ij}) \mid g, g_j \in E_G; 1 \leq i, j \leq \ell\}\).

First consider \(i \neq j\) such that \(g, g_j \in E_G\). Then there is a \(b_i \rightsquigarrow b_j\) path in \(M'\), namely, \(b_i \rightarrow h_{\phi(g)} \rightarrow c_{\phi(g)} \rightarrow (d_{\phi(g)}, a_{\phi(g)}) \rightarrow h_{\phi(g)} \rightarrow c_{\phi(g)} \rightarrow b_j\). Generalizing this and observing \(G\) is connected we can see any two terminals in \(B\) are strongly connected. Now consider two terminals \(f_{ij}\) and \(b_q\) such that \(1 \leq i, j, q \leq \ell\). The existence of the terminal \(f_{ij}\) implies \(g, g_j \in E_G\) and hence \(\phi(g_i), \phi(g_j) \in E_H\). There is a path in \(M'\) from \(f_{ij}\) to \(b_q\): use the path \(f_{ij} \rightsquigarrow d_{\phi(g)} \rightsquigarrow a_{\phi(g)} \rightarrow h_{\phi(g)} \rightarrow c_{\phi(g)} \rightarrow b_j\) followed by the \(b_j \rightsquigarrow b_q\) path (which was shown to exist above). Similarly there is a path in \(M'\) from \(b_q\) to \(f_{ij}\): use the \(b_q \rightsquigarrow b_\ell\) path (which was shown to exist above) followed by the path \(b_\ell \rightsquigarrow h_{\phi(g)} \rightarrow c_{\phi(g)} \rightarrow d_{\phi(g)} \rightarrow a_{\phi(g)} \rightarrow f_{ij}\). Hence each terminal in \(B\) can reach every terminal in \(F\) and vice versa. Finally consider any two terminals \(f_{ij}\) and \(f_{kl}\) in \(F\): the terminal \(f_{ij}\) can first reach \(b_i\) and we have seen above that \(b_i\) can reach any terminal in \(F\). This shows \(M'\) forms a solution for the SCSS instance.

**Lemma 6.3.** If the instance \((G^*, T^*)\) of SCSS has a solution of size \(\leq 3\ell + 10\lvert E_G\rvert\), then the instance \((G, H)\) of PSI answers YES.
Proof. Let $X$ be a solution of size $3\ell + 10|E_G|$ for the instance $(G^*, T^*)$ of SCSS. Consider a terminal $f_{ij} \in F$. The only out-neighbors of $f_{ij}$ are vertices from $D$, and hence $X$ must contain an edge $(f_{ij}, d_{vu})$ such that $v \in H_i$ and $u \in H_j$. However the only neighbor of $d_{vu}$ is $a_{vu}$, and hence $X$ has to contain this edge as well. Finally, $X$ must also contain one incoming edge into $f_{ij}$ since we desire strong connectivity. So for each terminal $f_{ij}$, we need three private dotted edges in the sense that every terminal in $F$ needs three such edges in any optimum solution. This uses up $6|E_G|$ of the budget since $|F| = 2|E_G|$. Referring to Figure 5, we can see any $f_{ij} \in F$ needs two private light edges in $X$: one edge coming out of some vertex in $A$ and some edge going into a vertex of $D$. This uses up $4|E_G|$ more from the budget leaving us with only $3\ell$ edges.

Consider $b_i$ for $i \in [\ell]$. First we claim that $X$ must contain at least three black edges for $b_i$ to have incoming and outgoing paths to the other terminals. The only outgoing edge from $b_i$ is to vertices of $H$, and hence we need to pick an edge $(v, b_i)$ such that $v \in H_i$. Since the only out-neighbor of $b_i$ is $c_v$, it follows that $X$ must pick this edge as well. Additionally, $X$ also needs to contain at least one incoming edge into $b_i$ to account for incoming paths from other terminals to $b_i$. So each $b_i$ needs to have at least three edges selected in order to have incoming and outgoing paths to other terminals. Moreover, all these edges are clearly private, i.e., different for each $b_i$. But as seen in the previous paragraph, our remaining budget was at most $3\ell$. Hence $X$ selects exactly three such edges for each $b_i$. We now claim that once $X$ contains the edges $(b_i, h_v)$ and $(h_v, c_v)$ such that $v \in H_i$ then $X$ must also contain the edge $(c_v, b_i)$. Suppose not, and for incoming towards $b_i$ the solution $X$ selects the edge $(c_v, b_i)$ for some $w \in H_i$ such that $w \neq v$. Then since $h_v$ is the only neighbor of $c_v$, the solution $X$ would be forced to select this edge as well. This implies that at least four edges have been selected for $b_i$, which is a contradiction. So for every $i \in [\ell]$, there is a vertex $v_i \in H_i$ such that the edges $(b_i, h_v), (h_v, c_v)$, and $(c_v, b_i)$ are selected in the solution for the SCSS instance. Further these are the only black edges in $X$ corresponding to $b_i$ (refer to Figure 5). It also follows for each $f_{ij} \in F$, the solution $X$ contains exactly three of the dotted edges (we argued above that each $f_{ij}$ needs three dotted edges, and the budget now implies that this is the maximum we can allow). Define $\phi : V_G \rightarrow V_H$ by $\phi(g_i) = v_i$ for each $i \in [\ell]$. Since $v_i \in H_i$ and the sets $H_1, H_2, \ldots, H_\ell$ form a disjoint partition of $V_H$, it follows that the function $\phi$ is an injection. Consider any edge $g_ig_j \in E_G$. We have seen above that the solution $X$ contains exactly three black edges per $f_{ij} \in F$. Suppose for $f_{ij} \in F$ the solution $X$ contains the edges $(f_{ij}, d_{vu}), (d_{vu}, a_{vu})$, and $(a_{vu}, f_{ij})$ for some $v \in H_i, u \in H_j$. The only incoming path for $f_{ij}$ is via $d_{vu}$. Also the only outgoing path from $b_i$ is via $c_v$. If $v_i \neq v$ then we will need two other dotted edges to reach $f_{ij}$, which is a contradiction since we have already picked the allocated budget of three such edges. Hence, $v_i = v$. Similarly, it follows that $v_j = u$. Finally, the existence of the vertex $d_{vu}$ implies $vu \in E_H$, i.e., $\phi(g_i)\phi(g_j) \in E_H$.  

6.1. Proof of Theorem 1.3. Finally, we are now ready to prove Theorem 1.3 which is restated below.

THEOREM 1.3. Under ETH, the edge-unweighted version of the SCSS problem cannot be solved in time $f(k) \cdot n^{o(k/\log k)}$, where $f$ is an arbitrary computable function, $k$ is the number of terminals, and $n$ is the number of vertices in the instance.

Proof. Lemmas 6.2 and 6.3 together give a parameterized reduction from PSI to SCSS. Observe that the number of terminals $k$ of the SCSS instance is $|B \cup F| = \infty$.
with a blue edge coming out of a

see later that \( \Delta = 5 \)

solution the paths between the terminals will be exactly the canonical paths. We will exactly \( \Delta \) (Thus the total weight of the first and the last blue edges on any canonical path is \( n \Delta \).)

d as viewed from top to bottom. They are named using magenta in Figure 6. Similarly to the

to 195.113.17.144. Redistribution subject to SIAM license or copyright; see http://www.siam.org/journals/ojsa.php

7. W[1]-hardness for DSN in planar DAGs. The main goal of this section is to prove Theorem 1.4 which is restated below.

Theorem 1.4. The edge-unweighted version of the DSN problem is W[1]-hard parameterized by the number \( k \) of terminal pairs, even when the input is restricted to planar directed acyclic graphs (planar DAGs). Moreover, there is no \( f(k) \cdot n^{o(k)} \) time algorithm for any computable function \( f \), unless the ETH fails.

Note that this shows that the \( n^{O(k)} \) algorithm of Feldman–Ruhl is asymptotically optimal. To prove Theorem 1.4, we reduce from the Grid Tiling problem introduced by Marx [57].

Consider an instance \( (k,n,\{S_{i,j} : 1 \leq i,j \leq k\}) \) of Grid Tiling. We now build an instance \( (G,T) \) of edge-weighted DSN as shown in Figure 6. Set \( T = \{(a_i,b_i) : (c_j,d_j) : i \in [k]\} \), i.e., we have 2\( k \) terminal pairs. We introduce \( k^2 \) red gadgets, where each gadget is an \( n \times n \) grid. Set the weight of each black edge to 2.

Definition 7.1. An \( a_i \rightsquigarrow b_i \) canonical path is a path from \( a_i \) to \( b_i \) which starts with a blue edge coming out of \( a_i \), then follows a horizontal path of black edges and finally ends with a blue edge going into \( b_i \).

Similarly, a \( c_j \rightsquigarrow d_j \) canonical path is a path from \( c_j \) to \( d_j \) which starts with a blue edge coming out of \( c_j \), then follows a vertically downward path of black edges and finally ends with a blue edge going into \( d_j \).

There are \( n \) edge-disjoint \( a_i \rightsquigarrow b_i \) canonical paths: let us call them \( P_1, P_2, \ldots, P_n \) as viewed from top to bottom. They are named using magenta in Figure 6. Similarly we call the canonical paths from \( c_j \) to \( d_j \) as \( Q_j^1, Q_j^2, \ldots, Q_j^m \) when viewed from left to right. For each \( i \in [k] \) and \( \ell \in [n] \) we assign a weight of \( \Delta(n+1-\ell) \) and \( \Delta \ell \) to the first and last black edges of \( P_\ell \), respectively. Similarly for each \( j \in [k] \) and \( \ell \in [n] \) we assign a weight of \( \Delta(n+1-\ell) \) and \( \Delta \ell \) to the first and last blue edges of \( Q_j^\ell \), respectively.

Thus the total weight of the first and the last blue edges on any canonical path is exactly \( \Delta(n+1) \). The idea is to choose \( \Delta \) large enough such that in any optimum solution the paths between the terminals will be exactly the canonical paths. We will see later that \( \Delta = 5n^2 \) will suffice for this purpose. Any canonical path consists of the following set of edges:

- Two blue edges (which sum up to \( \Delta(n+1) \));
- \( (k+1) \) black edges not inside the gadgets;
- \( (n-1) \) black edges inside each gadget.

Since the number of gadgets each canonical path visits is \( k \) and the weight of each black edge is 2, we have the total weight of any canonical path is \( \Delta(n+1)+2(k+1)+2k(n-1) \).
Intuitively the $k^2$ gadgets correspond to the $k^2$ sets in the Grid Tiling instance. Let us denote by $G_{i,j}$ the gadget which contains all vertices which lie on the intersection of any $a_i \rightsquigarrow b_i$ path and any $c_j \rightsquigarrow d_j$ path. If $(x,y) \in S_{i,j}$ then we color green the vertex in the gadget $G_{i,j}$ which is the unique intersection of the canonical paths $P_i^x$ and $Q_j^y$. Then we add a shortcut as shown in Figure 7. The idea is if both the $a_i \rightsquigarrow b_i$ path and $c_j \rightsquigarrow d_j$ path pass through the green vertex then the $a_i \rightsquigarrow b_i$ path can save a weight of 1 by using the green edge and a vertical edge to reach the green vertex, instead of paying a weight of 2 to use the horizontal edge reaching the green vertex. It is easy to see that there is a solution (without using green edges) for the DSN instance of weight $B^* = 2k(\Delta(n+1) + 2(k+1) + 2k(n-1))$: each terminal pair just uses a canonical path and these canonical paths are pairwise edge disjoint.

The following assumption will be helpful in handling some of the border cases of the gadget construction. We may assume that $1 < \min\{x,y\}$ holds for every $(x,y) \in S_{i,j}$; indeed, we can increase $n$ by one and replace every $(x,y)$ by $(x+1,y+1)$ without changing the problem. Hence, no green vertex can be in the first row or first column of any gadget. Combining this fact with the orientation of the edges we get the only gadgets which can intersect any $a_i \rightsquigarrow b_i$ path are $G_{i,1}, G_{i,2}, \ldots, G_{i,k}$. Similarly the only gadgets which can intersect any $c_j \rightsquigarrow d_j$ path are $G_{1,j}, G_{2,j}, \ldots, G_{k,j}$. This completes the construction of the instance $(G,T)$ of DSN.

Lemmas 7.2 and 7.6 below prove that the reduction described above is indeed a correct reduction from Grid Tiling to DSN.

**Lemma 7.2.** If the instance $(k,n,\{S_{i,j} : 1 \leq i,j \leq k\})$ of Grid Tiling has a solution then the instance $(G,T)$ of DSN has a solution of weight at most $B^* - k^2$. 
Grid Tiling

The idea is if both the edges \((y, v)\) and \((u, v)\) were being used initially then now we can save a weight of 1 by making the horizontal path choose \((u, x)\) and then we get \((x, v)\) for free, as it is already being used by the vertical canonical path.

\begin{figure}[h]
\centering
\includegraphics[width=0.2	extwidth]{fig7.png}
\caption{Let \(u, v\) be two consecutive vertices on the canonical path \(P_i\). Let \(v\) be on the canonical path \(Q_j\) and let \(y\) be the vertex preceding it on this path. If \(v\) is a green vertex then we subdivide the edge \((y, v)\) by introducing a new vertex \(x\) and adding two edges \((y, x)\) and \((x, v)\) of weight 1. We also add an edge \((u, x)\) of weight 1.}
\end{figure}

Proof. For each \(1 \leq i, j \leq k\) let \(s_{i,j} \in S_{i,j}\) be the entry in the solution of the Grid Tiling instance. Therefore for every \(i \in k\) we know that each of the \(k\) entries \(s_{i,1}, s_{i,2}, \ldots, s_{i,k}\) have the same first coordinate \(a_i\). Similarly for every \(j \in [k]\) each of the \(k\) vertices \(s_{1,j}, s_{2,j}, \ldots, s_{k,j}\) has the same second coordinate \(\gamma_j\). For each \(j \in [k]\) we use the canonical path \(Q_j\) to satisfy the terminal for \((c_j, d_j)\). For each \(i \in [k]\), we essentially use the canonical path \(P_i\) with the following modifications: for each \(j \in [k]\), take the shortcut green edge (as shown in Figure 7) when we encounter the green vertex (this is guaranteed to happen since \((a_i, \gamma_j) = s_{i,j} \in S_{i,j}\)) in \(G_{i,j}\) at the intersection of the canonical paths \(P_i\) and \(Q_j\). Hence, overall we save a total of \(k^2\): a saving of one per gadget. Thus, we have produced a solution for the instance \((G, T)\) of weight \(2k(\Delta(n + 1) + 2(k + 1) + 2k(n - 1)) - k^2 = B' - k^2\).

We now prove the other direction which is more involved. First we show some preliminary claims.

Claim 7.3. Any optimum solution for \((G, T)\) contains a \(c_j \rightsquigarrow d_j\) canonical path for each \(j \in [k]\).

Proof. Suppose to the contrary that there is an optimum solution \(N\) for \((G, T)\) which does not contain a canonical \(c_j \rightsquigarrow d_j\) path for some \(j \in [k]\). From the orientation of the edges, we know that there is a \(c_j \rightsquigarrow d_j\) path in \(N\) that starts with the blue edge from \(Q_j^i\) and ends with a blue edge from \(Q_j^\ell\) for some \(\ell > \ell\). We create a new set of edges \(N'\) from \(N\) as follows:

\begin{itemize}
\item Add all those edges of \(Q_j^\ell\) which were not present in \(N\). In particular, we add the last blue edge of \(Q_j^\ell\) since \(\ell > \ell\).
\item Delete the last blue edge of \(Q_j^\ell\).
\end{itemize}

It is easy to see that \(N'\) is also a solution for \((G, T)\): this is because \(N'\) contains the canonical path \(Q_j^i\) to satisfy the pair \((c_j, d_j)\), and the last (blue) edge of any \(c_j \rightsquigarrow d_j\) canonical path cannot be on any \(a_i \rightsquigarrow b_i\) path for any \(i \in [k]\). Changing the last blue edge saves us \((\ell' - \ell)\Delta \leq \Delta = 5m^2\). However we have to be careful since we added some edges to the solution. But these edges are the internal (black) edges of \(Q_j^i\), and their weight is \(\leq 2(k + 1) + 2k(n - 1) = 2kn + 2 < 5m^2 = \Delta\) since \(1 \leq k \leq n\). Therefore we are able to create a new solution \(N'\) whose weight is less than that of an optimum solution \(N\), which is a contradiction.

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Definition 7.4. An \( a_i \rightsquigarrow b_i \) path is called an almost canonical path if its first and last edges are blue edges from the same \( a_i \rightsquigarrow b_i \) canonical path.

Hence, an \( a_i \rightsquigarrow b_i \) almost canonical path looks very similar to an \( a_i \rightsquigarrow b_i \) canonical path, except it can replace some of the horizontal blue edges by green edges and vertical black edges as shown in Figure 7. However, note that by definition, an almost canonical path must however end on the same horizontal level on which it began. The proof of the next claim is very similar to that of Claim 7.3.

Claim 7.5. Any optimum solution for DSN contains an \( a_i \rightsquigarrow b_i \) almost canonical path for every \( i \in [k] \).

Proof. Suppose to the contrary that there is an optimum solution \( N \) which does not contain an almost canonical \( a_i \rightsquigarrow b_i \) path for some \( i \in [k] \). Hence, the \( a_i \rightsquigarrow b_i \) path in \( N \) starts and ends at different levels. From the orientation of the edges, we know that there is an \( a_i \rightsquigarrow b_i \) path in the optimum solution that starts with the blue edge from \( P_i^l \) and ends with a blue edge from \( P_i^{l'} \) for some \( l' > l \) (note that the construction in Figure 7 does not allow any \( a_i \rightsquigarrow b_i \) path to climb onto an upper level).

We create a new set of edges \( N' \) from \( N \) as follows:

- Add all those edges of \( P_i^{l'} \) which were not present in \( N \). Note that, in particular, we add the last blue edge of \( P_i^{l'} \) since \( l' > l \).
- Delete the last blue edge of \( P_i^l \).

It is easy to see that \( N' \) is also a solution for \( (G, T) \): this is because \( N' \) contains the canonical path \( P_i^l \) to satisfy the pair \((a_i, b_i)\), and the last (blue) edge of any \( a_i \rightsquigarrow b_i \) canonical path cannot be on any \( c_j \rightsquigarrow d_j \) path for any \( j \in [k] \). Changing the last edge saves us \((l' - l)\Delta \leq \Delta = 5n^2 \). But we have to careful since we also added some edges to the solution. The total weight of edges added is \( \leq 2(k + 1) + 2k(n - 1) = 2kn + 2 < 5n^2 = \Delta \) since \( 1 \leq k \leq n \). So we can create a new solution \( N' \) whose weight is less than that of an optimum solution \( N \), which is a contradiction.

Lemma 7.6. If the instance \((G, T)\) of DSN has a solution of weight at most \( B^* - k^2 \), then the instance \((k, n, \{S_{i,j} : 1 \leq i,j \leq k\})\) of Grid Tiling has a solution.

Proof. Consider any optimum solution \( X \). By Claims 7.3 and 7.5 we know that \( X \) has an \( a_i \rightsquigarrow b_i \) almost canonical path and a \( c_j \rightsquigarrow d_j \) canonical path for every \( 1 \leq i,j \leq k \). Moreover these sets of 2k paths form a solution for DSN. Since any optimum solution is minimal, \( X \) is the union of these 2k paths: one for each terminal pair. For each \( i,j \in [k] \) let the \( a_i \rightsquigarrow b_i \) almost canonical path in \( X \) be \( T_i^{\alpha_i} \) and the \( c_j \rightsquigarrow d_j \) canonical path in \( X \) be \( Q_j^{\gamma_j} \).

The \( a_i \rightsquigarrow b_i \) almost canonical path \( T_i^{\alpha_i} \) and \( c_j \rightsquigarrow d_j \) canonical path \( Q_j^{\gamma_j} \) in \( X \) intersect in a unique vertex in the gadget \( G_{i,j} \). If each \( a_i \rightsquigarrow b_i \) path was canonical instead of almost canonical, then the weight of \( X \) would have been exactly \( B^* \). However we know that the weight of \( X \) is at most \( B^* - k^2 \). It is easy to see any \( a_i \rightsquigarrow b_i \) almost canonical path and any \( c_j \rightsquigarrow d_j \) canonical path can have at most one edge in common: the edge which comes vertically downwards into the green vertex (see Figure 7). There are \( k^2 \) gadgets, and there is at most one edge per gadget which is used for two paths in \( X \). Hence for each gadget \( G_{i,j} \) there is exactly one edge which is used by both the \( a_i \rightsquigarrow b_i \) almost canonical path and the \( c_j \rightsquigarrow d_j \) canonical path in \( X \). So the endpoint of each of these common edges must be a green vertex, i.e., \((\alpha_i, \gamma_j) \in S_{i,j} \) for each \( i,j \in [k] \).
7.1. Proof of Theorem 1.4. Finally, we are now ready to prove Theorem 1.4 which is restated below.

**Theorem 1.4.** The edge-unweighted version of the DSN problem is \( W[1] \)-hard parameterized by the number \( k \) of terminal pairs, even when the input is restricted to planar directed acyclic graphs (planar DAGs). Moreover, there is no \( f(k) \cdot n^{o(k)} \) time algorithm for any computable function \( f \), unless the ETH fails.

**Proof.** Given an instance \( (k, n, \{S_{i,j} : 1 \leq i,j \leq k\}) \) of **Grid Tiling**, we use the reduction described earlier in this section to build an instance \((G, T)\) of edge-weighted DSN (see Figure 6 for an illustration). It is easy to see that the total number of vertices in \( G \) is \( O(n^2k^2) \) and moreover can be constructed in \( \text{poly}(n, k) \) time. Each grid is planar (green shortcut edges do not destroy planarity), and the grids are arranged again in a gridlike manner. Figure 6 actually gives a planar embedding of \( G \).

Moreover, it is not hard to observe that \( G \) is a DAG.

It is known [23, Theorem 14.28] that \( k \times k \) **Grid Tiling** is \( W[1] \)-hard parameterized by \( k \), and under ETH cannot be solved in \( f(k) \cdot n^{o(k)} \) for any computable function \( f \). Combining the two directions from Lemmas 7.2 and 7.6, we get a parameterized reduction from \( k \times k \) **Grid Tiling** to an instance of DSN which is a planar DAG and has \( O(k) \) terminal pairs. Hence, it follows that DSN on planar DAGs is \( W[1] \)-hard and under ETH cannot be solved in \( f(k) \cdot n^{o(k)} \) time for any computable function \( f \).

Note that Theorem 1.4 shows that the \( n^{O(k)} \) algorithm of Feldman–Ruhl [35] for DSN is asymptotically optimal.

**Appendix A.** Vertex-unweighted versions are more general than edge-weighted versions with integer weights.

In this section, for both the SCSS and DSN problems we show that the edge-weighted version (with polynomially bounded integer weights) can be solved using the vertex-unweighted version. Hence all our hardness results from Theorems 1.2, 1.3, and 1.4 hold for the vertex-(un)weighted versions as well.

We give a formal proof for the DSN problem; the proof for the SCSS problem is similar. Consider an instance \( I_1 = (G, T) \) of edge-weighted DSN with integer weights, where \( T = \{(s_i, t_i) \mid i \in [k]\} \). Replace each edge of weight \( \ell \) by \( n\ell \) internal vertices, where \( |G| = n \). Let the new graph be \( G' \). Consider the instance \( I_2 \) of the vertex-unweighted version where the set of terminals is the same as in \( I_1 \).

**Theorem A.1.** The instance \( I_1 \) of edge-weighted DSN has a solution of weight at most \( Cn \) if and only if the instance \( I_2 \) of vertex-unweighted DSN has a solution with at most \( Cn + n \) vertices.

**Proof.** Suppose there is a solution \( E_1 \) for \( I_1 \) of weight at most \( C \). For each edge in \( E_1 \) pick all its internal vertices and two endpoints in \( E_2 \). Clearly \( E_2 \) is a solution for \( I_2 \). The number of vertices in \( E_2 \) is \( Cn + \gamma \), where \( \gamma \) is the number of vertices of \( G \) incident to the edges in \( E_1 \). Since \( \gamma \leq n \) we are done.

Suppose there is a (vertex-minimal) solution \( E_2 \) for \( I_2 \) having at most \( Cn + n \) vertices. For any edge \( e \in G \) of weight \( e \) we need to pick all the \( cn \) internal vertices (plus the two endpoints of \( e \)) in \( E_2 \) if we actually want to use \( e \) in a solution for \( I_1 \). So for every edge \( e \in E \) we know that \( E_2 \) contains either all or none of the internal vertices obtained after splitting up \( e \) according to its weight in \( G \). Let the set of edges of \( G \) all of whose internal vertices are in \( E_2 \) be \( E_1 = \{e_1, e_2, \ldots, e_r\} \) and their weights be \( c_1, c_2, \ldots, c_r \), respectively. Since \( E_2 \) is a solution for \( I_2 \) it follows that \( E_1 \)
is a solution for $I_1$. Let $S$ be the union of the set of endpoints of the edges in $E_1$. Therefore $Cn + n \geq |S| + n(\sum_{i=1}^r c_i)$. Since $|S| \geq 1$ we have $C \geq \sum_{i=1}^r c_i$, i.e., $E_1$ has weight at most $C$. \qed

Note that the above reduction works even in the case when the edges have zero weight:\footnote{We mention this explicitly because some of the reductions in this paper do have edges with zero weight.} in this case we simply won’t be adding any internal vertices.

Appendix B. Treewidth and minors.

**Definition B.1** (treewidth). Let $G$ be a given undirected graph. Let $T$ be a tree and $B : V(T) \to 2^{V(G)}$. The pair $(T,B)$ called a tree decomposition of an undirected graph $G$ is a tree $T$ in which every vertex $x \in V(T)$ has an assigned set of vertices $B_x \subseteq V(G)$ (called a bag) such that the following properties are satisfied:

- (P1): $\bigcup_{x \in V(T)} B_x = V(G)$.
- (P2): For each $\{u,v\} \in E(G)$, there exists an $x \in V(T)$ such that $u,v \in B_x$.
- (P3): For each $v \in V(G)$, the set of vertices of $T$ whose bags contain $v$ induce a connected subtree of $T$.

The width of the tree decomposition $(T,B)$ is $\max_{x \in V(T)} |B_x| - 1$. The treewidth of a graph $G$, usually denoted by $\text{tw}(G)$, is the minimum width over all tree decompositions of $G$.

**Definition B.2** (minor). Let $G,H$ be undirected graphs. Then $H$ is called a minor of $G$ if $H$ can be obtained from $G$ by deleting edges, deleting vertices, and by contracting edges.

**Definition B.3** (subdivision). Let $G$ be an undirected graph. An edge $e = u-v$ is subdivided by adding a new vertex $w$ and the edges $u-w$ and $v-w$. An undirected graph $H$ is called a subdivision of $G$ if $H$ can be obtained from $G$ by subdividing edges of $G$.

**Lemma B.4.** Subdivisions of outerplanar graphs have treewidth at most 2.

**Proof.** Outerplanar graphs are known to be a subclass of series parallel graphs and, hence, have treewidth at most 2. To prove this lemma, it is enough to show that subdividing one edge of an outerplanar does not increase the treewidth. Let $G$ be an outerplanar graph, and $(T,B)$ be a tree decomposition of $G$ of width at most 2. Let $e = u-v$ be an edge in $G$ which is subdivided by adding a vertex $w$ and the edges $u-w$ and $v-w$. We now build a tree decomposition for the resulting graph $G'$. We add only one vertex to $V(T)$: by property (P2), there exists $t \in V(T)$ such that $u,v \in B_t$. Add a new vertex $t'$ and set $B_{t'} = \{u,v,w\}$. Make $t'$ adjacent only to $t$. It is easy to check that $V(T) \cup \{t'\}$ is a tree decomposition for $G'$ of treewidth at most 2. \qed

**Acknowledgment.** We would like to thank the anonymous reviewers for their detailed and helpful comments which have helped to improve the presentation.

REFERENCES


TIGHT BOUNDS FOR PLANAR SCSS (AND EXTENSIONS)


Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
[20] R. Curticapean, H. Dell, and D. Marx, Homomorphisms are a good basis for
counting small subgraphs, in Proceedings of the 49th Annual ACM SIGACT

[21] R. Curticapean and D. Marx, Complexity of counting subgraphs: Only the
boundedness of the vertex-cover number counts, in Proceedings of the 55th IEEE
Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia,

[22] R. Curticapean and M. Xia, Parameterizing the permanent: Genus, apices, minors,
evaluation mod 2k, in Proceedings of the IEEE 56th Annual Symposium on
Foundations of Computer Science, FOCS 2015, Berkeley, CA, IEEE, Piscataway, NJ,

[23] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk,
M. Pilipczuk, and S. Saurabh, Parameterized Algorithms, Springer, Cham,
Switzerland, 2015, https://doi.org/10.1007/978-3-319-21275-3.

parameterized algorithms on bounded-genus graphs and H-minor-free graphs, J. ACM,


[26] E. D. Demaine and M. T. Hajiaghayi, Graphs excluding a fixed minor have grids
as large as treewidth, with combinatorial and algorithmic applications through
bidimensionality, in Proceedings of the Sixteenth Annual ACM-SIAM Symposium on
http://dl.acm.org/citation.cfm?id=1070432.1070528.

[27] E. D. Demaine, F. V. Fomin, M. T. Hajiaghayi, and P. N. Klein, Node-weighted
Steiner tree and group Steiner tree in planar graphs, ACM Trans. Algorithms, 10


Parameterized approximation schemes for Steiner trees with small number of
Steiner vertices, in Proceedings of the 35th Symposium on Theoretical Aspects of
Computer Science, STACS 2018, Caen, France, Wadern, Germany, Schloss

with respect to the number of terminals, in 36th International Symposium on
Theoretical Aspects of Computer Science, STACS 2019, Berlin, Germany, Wadern,

for Steiner forest in planar graphs, in Proceedings of the Twenty-Third Annual ACM-SIAM
Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, SIAM, Philadelphia, 2012,

What makes equitable connected partition easy, in Parameterized and Exact Computation,
4th International Workshop, IWPEC 2009, Copenhagen, Denmark, 2009, Revised Selected

[34] D. Epstein and D. Lokshtanov, The parameterized complexity of finding point sets with
hereditary properties, in Proceedings of the 13th International Symposium on Parameterized
and Exact Computation, IPEC 2018, Helsinki, Finland, Wadern, Germany, Schloss

[35] J. Feldman and M. Ruhl, The directed Steiner network problem is tractable for a constant
S0097539704441241.


Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.


Parameterized Approximation Algorithms for Bidirected Steiner Network Problems

Rajesh Chitnis\textsuperscript{1}
University of Warwick, UK
rajeshchitnis@gmail.com

Andreas Emil Feldmann\textsuperscript{2}
Charles University, Prague, Czechia
feldmann.a.e@gmail.com

Pasin Manurangsi\textsuperscript{3}
University of California, Berkeley, USA
pasin@berkeley.edu

Abstract
The Directed Steiner Network (DSN) problem takes as input a directed edge-weighted graph $G = (V, E)$ and a set $D \subseteq V \times V$ of $k$ demand pairs. The aim is to compute the cheapest network $N \subseteq G$ for which there is an $s \rightarrow t$ path for each $(s,t) \in D$. It is known that this problem is notoriously hard as there is no $k^{1/4-o(1)}$-approximation algorithm under Gap-ETH, even when parameterizing the runtime by $k$ [Dinur & Manurangsi, ITCS 2018]. In light of this, we systematically study several special cases of DSN and determine their parameterized approximability for the parameter $k$.

For the bi-DSN\textsuperscript{PLANAR} problem, the aim is to compute a planar optimum solution $N \subseteq G$ in a bidirected graph $G$, i.e. for every edge $uv$ of $G$ the reverse edge $vu$ exists and has the same weight. This problem is a generalization of several well-studied special cases. Our main result is that this problem admits a parameterized approximation scheme (PAS) for $k$. We also prove that our result is tight in the sense that (a) the runtime of our PAS cannot be significantly improved, and (b) it is unlikely that a PAS exists for any generalization of bi-DSN\textsuperscript{PLANAR}, unless FPT=W[1]. Additionally we study several generalizations of bi-DSN\textsuperscript{PLANAR} and obtain upper and lower bounds on obtainable runtimes parameterized by $k$.

One important special case of DSN is the Strongly Connected Steiner Subgraph (SCSS) problem, for which the solution network $N \subseteq G$ needs to strongly connect a given set of $k$ terminals. It has been observed before that for SCSS a parameterized 2-approximation exists when parameterized by $k$ [Chitnis et al., IPEC 2013]. We show a tight inapproximability result: under Gap-ETH there is no $(2-\epsilon)$-approximation algorithm parameterized by $k$ (for any $\epsilon > 0$). To the best of our knowledge, this is the first example of a W[1]-hard problem admitting a non-trivial parameterized approximation factor which is also known to be tight! Additionally we show that when restricting the input of SCSS to bidirected graphs, the problem remains NP-hard but becomes FPT for $k$.

2012 ACM Subject Classification Theory of computation → Routing and network design problems, Theory of computation → Fixed parameter tractability

Keywords and phrases Directed Steiner Network, Strongly Connected Steiner Subgraph, Parameterized Approximations, Bidirected Graphs, Planar Graphs

\textsuperscript{1} Supported by ERC grant 2014-CoG 647557. Part of this work was done while at Weizmann Institute of Science, Israel and supported by Israel Science Foundation grant #897/13

\textsuperscript{2} Supported by the Czech Science Foundation GAČR (grant #17-10090Y), and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004).

\textsuperscript{3} This work was done while the author was visiting Weizmann Institute of Science.

© Rajesh Chitnis, Andreas Emil Feldmann, and Pasin Manurangsi; licensed under Creative Commons License CC-BY

26th Annual European Symposium on Algorithms (ESA 2018)
Editors: Yossi Azar, Hannah Bast, and Grzegorz Herman; Article No. 20; pp. 20:1–20:16
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany
1 Introduction

In this paper we study the Directed Steiner Network (DSN) problem\(^4\), in which a directed edge-weighted graph \(G = (V, E)\) is given together with a set of \(k\) demands \(D = \{(s_i, t_i)\}_{i=1}^{k} \subseteq V \times V\). The aim is to compute a minimum cost (in terms of edge weights) network \(N \subseteq G\) containing a directed \(s_i \rightarrow t_i\) path for each \(i \in \{1, \ldots, k\}\). This well-studied problem has applications in network design [38], and for instance models the setting where nodes in a radio or ad-hoc wireless network connect to each other unidirectionally [10, 57].

The DSN problem is notoriously hard. First of all, it is NP-hard, and one popular way to handle NP-hard problems is to efficiently compute an \(\alpha\)-approximation, i.e., a solution that is guaranteed to be at most a factor \(\alpha\) worse than the optimum. For this paradigm we typically demand that the algorithm computing such a solution runs in polynomial time in the input size \(n = |V|\). However for DSN it is known that even computing an \(O(2^{\log^{1-\epsilon} n})\)-approximation is not possible [18] in polynomial time, unless \(\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})\). It is possible to obtain approximation factors \(O(n^{2/3+\epsilon})\) and \(O(k^{1/2+\epsilon})\) though [3, 9, 25]. For settings where the number \(k\) of demands is fairly small, one may aim for algorithms that only have a mild exponential runtime blow-up in \(k\), i.e., a runtime of the form \(f(k) \cdot n^{O(1)}\), where \(f(k)\) is some function independent of \(n\). If an algorithm computing the optimum solution with such a runtime exists for a computable function \(f(k)\), then the problem is called fixed-parameter tractable (FPT) for parameter \(k\). However it is unlikely that DSN is FPT for this well-studied parameter, as it is known to be \(W[1]\)-hard [31] for \(k\). In fact one can show [14, 22] that under the Exponential Time Hypothesis (ETH) there is no algorithm computing the optimum in time \(f(k) \cdot n^{o(k)}\) for any function \(f(k)\) independent of \(n\). ETH assumes that there is no \(2^{o(n)}\) time algorithm to solve 3SAT [33, 34]. The best we can hope for is therefore a so-called XP-algorithm computing the optimum in time \(n^{O(k)}\), and this was also shown to exist by Feldman and Ruhl [24].

None of the above algorithms for DSN seem satisfying though, either due to slow runtimes or large approximation factors, and this is hardly surprising given the problem’s inherent complexity. To circumvent the hardness of the problem, one may aim for parameterized approximations, which have recently received increased attention for various problems (see e.g. [5, 8, 11, 13, 23, 26, 42, 44, 46, 49, 59, 62, 21, 4, 37]). In this paradigm an \(\alpha\)-approximation is computed in time \(f(k) \cdot n^{O(1)}\) for parameter \(k\), where \(f(k)\) again is a computable function independent of \(n\). Unfortunately, a recent result by Dinur and Manurangsi [17]\(^5\) excludes significant improvements over the known polynomial time approximation algorithms [3, 9, 25], even if allowing a runtime parameterized in \(k\). More specifically, no \(k^{1/4 - o(1)}\)-approximation

\(^4\) Also sometimes called Directed Steiner Forest. Note however that in contrast to the undirected Steiner Forest problem, an optimum solution to DSN is not necessarily a forest.

\(^5\) In a previous version of this work, we showed that no \(k^{o(1)}\)-approximation is possible for DSN in time \(f(k) \cdot n^{O(1)}\). This result in now subsumed by [17].
is possible in time \( f(k) \cdot n^{O(1)} \) for any function \( f(k) \) under the \textit{Gap Exponential Time Hypothesis (Gap-ETH)}\(^6\), which postulates that there exists a constant \( \varepsilon > 0 \) such that no (possibly randomized) algorithm running in \( 2^{o(n)} \) time can distinguish whether it is possible to satisfy all or at most a \((1 - \varepsilon)\)-fraction of clauses of any given 3SAT formula [16, 48].

Given these hardness results, the main question we explore is: what approximation factors and runtimes are possible for special cases of DSN when parametrizing by \( k \)? There are two types of standard special cases that are considered in the literature:

- Restricting the input graph \( G \) to some special graph class. A typical assumption for instance is that \( G \) is planar.\(^7\)
- Restricting the pattern of the demands in \( D \). For example, one standard restriction is to have a set \( R \subseteq V \) of terminals, a fixed root \( r \in R \), and demand set \( D = \{(r, t) \mid t \in R\} \), which is the well-known \textit{Directed Steiner Tree (DST)} problem.

In fact, an optimum solution to the DST problem is an arborescence (hence the name), i.e., it is planar. Thus if an algorithm is able to compute (an approximation to) the cheapest planar DSN solution in an otherwise unrestricted graph, it can be used for both the above types of restrictions: it can of course be used if the input graph is planar as well, and it can also be used if the demand pattern implies that the optimum must be planar. Taking the structure of the optimum solution into account has been a fruitful approach leading to several results on related problems, both for approximation and fixed-parameter tractability, from which we also draw some of the inspiration for our results (cf. Section 1.2). A main focus of our work is to systematically explore the influence of the structure of optimum solutions on the complexity of the DSN problem. Formally, fixing a class \( K \) of graphs, we define the DSN\(_{K} \) problem, which asks for an optimum solution network \( N \subseteq G \) for \( k \) given demands such that \( N \in K \). The DSN\(_{K} \) problem has been implicitly studied in several results before for various classes \( K \), in particular when \( K \) contains either planar graphs, or graphs of bounded treewidth\(^8\) (cf. Table 1).

Another special case we consider is when the input graph \( G \) is \textit{bidirected}, i.e., for every edge \( uv \) of \( G \) the reverse edge \( vu \) exists in \( G \) as well and has the same weight as \( uv \). This naturally captures the problem variant between the notoriously hard DSN problem on directed graphs and its undirected counterpart the \textit{Steiner Forest (SF)} problem. As the former does not allow any \( \leq 1/4 - o(1) \)-approximation in time \( f(k) \cdot n^{O(1)} \) under Gap-ETH [17], while the latter is FPT [53, 27, 19] for parameter \( k \), it is interesting to ask what happens between these two extremes. Bidirected graphs also model the realistic setting [10, 57, 61, 43] when the cost of transmitting from a node \( u \) to a node \( v \) in a wireless network is the same in both directions, which for instance happens if the nodes all have the same transmitter model.

We meticulously study several special cases of DSN resulting from the above restrictions, and prove matching upper and lower bounds on runtimes parameterized by \( k \). We now give a brief overview of the studied problems emphasizing the main insights, and refer to Section 1.1 for a detailed exposition of our obtained results.

\textbf{Bi-DSN\(_{\text{PLANAR}} \), i.e., the DSN\(_{K} \) problem on bidirected inputs, where \( K \) is the class of planar graphs:} For this problem we present our main result, which is that bi-DSN\(_{\text{PLANAR}} \) admits a \textit{parameterized approximation scheme (PAS)}, i.e., an algorithm that for any \( \varepsilon > 0 \) computes

---

\(^6\) Gap-ETH follows from ETH given other standard conjectures, such as the existence of linear sized PCPs or exponentially-hard locally-computable one-way functions. See [8, 2] for more details.

\(^7\) A directed graph is planar if the underlying undirected graph is.

\(^8\) Here the \textit{undirected} treewidth is meant, i.e., the treewidth of the underlying undirected graph.
Parameterized Approximations for Bidirected Steiner Networks

a \((1 + \varepsilon)\)-approximation in \(f(\varepsilon, k) \cdot n^{\Theta(\varepsilon)}\) time for some functions \(f\) and \(g\). We also prove that, unless \(\text{FPT} = \text{W}[1]\), no \textit{efficient parameterized approximation scheme (EPAS)} exists, i.e., there is no algorithm computing a \((1 + \varepsilon)\)-approximation in \(f(\varepsilon, k) \cdot n^{O(1)}\) time for any function \(f\). Thus the runtime of our algorithm cannot be significantly improved.

\textbf{Bi-DSN}, i.e., the DSN problem on bidirected inputs: The above PAS for the restricted \textbf{Bi-DSN} _\text{Planar} problem begs the question of whether a PAS also exists for any more general problems, such as \textbf{Bi-DSN}. However we prove that \textbf{Bi-DSN} does not admit a PAS under Gap-ETH. At the same time it is not too hard to obtain constant approximations in parameterized or polynomial time, given known algorithms for SF. When aiming for optimum solutions however, surprisingly we can show that \textbf{Bi-DSN} is almost as hard as DSN (with almost-matching runtime lower bound under ETH). Thus the complexity of the in-between bidirected setting resembles that of the directed setting in terms of FPT algorithms, while in terms of approximations it is more similar to the undirected setting.

Apart from the DST problem, another well-studied special case of DSN with restricted demands is when the demand pairs form a cycle, i.e., we are given a set \(R = \{t_1, \ldots, t_k\}\) of \(k\) terminals and the set of demands is \(D = \{(t_i, t_{i+1})\}_{i=1}^k\) where \(t_{k+1} = t_1\). Since this implies that any optimum solution is strongly connected, this problem is accordingly known as the \textit{Strongly Connected Steiner Subgraph (SCSS)} problem. In contrast to DST, it is implicit from [31] (by a reduction from the \textit{Clique} problem) that optimum solutions to SCSS do not belong to any minor-closed graph class. Thus SCSS is not easily captured by some DSN\(K\) problem for a restricted class \(K\). Nevertheless it is still possible to exploit the structure of the optimum solution to SCSS, which results in the following findings.

\textbf{SCSS}: It is known that a 2-approximation is obtainable [13] when parametrizing by \(k\). We prove that the factor of 2 is best possible under Gap-ETH. To the best of our knowledge, this is the first example of a \(\text{W}[1]\)-hard problem having a parameterized approximation algorithm with non-trivial approximation factor (in this case 2), which is also known to be tight!

\textbf{Bi-SCSS}, i.e., the SCSS problem on bidirected inputs: As for \textbf{Bi-DSN}, one might think that \textbf{Bi-SCSS} is easily solvable via its undirected version, i.e., the well-known \textit{Steiner Tree (ST)} problem, which is \(\text{FPT}\) [53, 19] for parameter \(k\). However, it is not the case that simply taking an optimum undirected solution twice in a bidirected graph will produce a (near-)optimum solution to \textbf{Bi-SCSS} (see Figure 1). Nevertheless we prove that \textbf{Bi-SCSS} is \(\text{FPT}\) for parameter \(k\) as well, while also being \(\text{NP-hard}\). Our algorithm is non-trivial and does not apply any methods used for undirected graphs. To the best of our knowledge, bidirected inputs are the first example where SCSS remains NP-hard but turns out to be \(\text{FPT}\) parameterized by \(k\)!

Thus in contrast to \textbf{Bi-DSN}, the complexity of the in-between \textbf{Bi-SCSS} problem resembles that of the undirected variant (the ST problem) rather than the directed version (the SCSS problem).

### 1.1 Our results

Due to space constraints, almost all proofs of the following theorems are deferred to the full version of the paper [12].

\textbf{Bidirected inputs with planar solutions}. Our main theorem implies the existence of a PAS for \textbf{Bi-DSN}\_\text{Planar}, where the parameter is the number \(k\) of demands.

**Theorem 1.** For any \(\varepsilon > 0\), there is a \(\max\left\{\frac{2^k \varepsilon^{2(1/\varepsilon)}}{\log k}, n^{2(1/\varepsilon)}\right\}\) time algorithm for \textbf{Bi-DSN}_\text{Planar}, that computes a \((1 + \varepsilon)\)-approximation.
Figure 1 A bi-SCSS instance where all vertices are terminals. Left: Black edges show a solution which takes an undirected optimum twice. Right: The actual optimum solution is shown in black.

As bi-DSN_{PLANAR} is a rather restricted special case of DSN, one may at this point rightfully ask: Should it not be possible to obtain better runtimes and/or should it not be possible to even compute the optimum solution when parametrizing by $k$? And could it not be that a similar result is true in more general settings, when for instance the input is bidirected but the optimum is not restricted to a planar graph? We prove that both questions can be answered in the negative.

First off, it is not hard to prove that a polynomial time approximation scheme (PTAS) is not possible for bi-DSN_{PLANAR}, i.e., it is necessary to parametrize by $k$ in Theorem 1. This is implied by the following result, since (as mentioned before) a PTAS for bi-DSN_{PLANAR} would also imply a PTAS for bi-DST, i.e., the DST problem on bidirected input graphs.

Theorem 2. The bi-DST problem is APX-hard.

One may wonder however, whether parametrizing by $k$ doesn’t make the bi-DSN_{PLANAR} problem FPT, so that approximating the planar optimum as in Theorem 1 would in fact be unnecessary. Furthermore, even if it is necessary to approximate, one may ask whether the runtime given in Theorem 1 can be improved. In particular, note that the runtime we obtain in Theorem 1 is similar to that of a PTAS, which has a runtime of the form $f(k, \varepsilon) \cdot n^{O(1)}$, i.e., we would like to treat $\varepsilon$ as a parameter as well. The following theorem shows that both approximating and runtime dependence on $\varepsilon$ are in fact necessary in Theorem 1.

Theorem 3. The bi-DSN_{PLANAR} problem is W[1]-hard parameterized by $k$. Moreover, under ETH, for any computable functions $f(k)$ and $f(k, \varepsilon)$, and parameters $k$ and $\varepsilon > 0$, the bi-DSN_{PLANAR} problem has no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm to compute the optimum solution, and has no $f(k, \varepsilon) \cdot n^{o(\sqrt{k})}$ time algorithm to compute a $(1 + \varepsilon)$-approximation.

It stands out that to compute optimum solutions, this theorem rules out runtimes for which the dependence of the exponent of $n$ is $o(\sqrt{k})$, while for the general DSN problem, as mentioned above, the both necessary and sufficient dependence of the exponent is linear in $k$ [24, 14]. Could it be that bi-DSN_{PLANAR} is just as hard as DSN when computing optimum solutions? The answer is no, as the next theorem shows.

Theorem 4. There is a $2^{O(k^{3/2} \log k)} \cdot n^{O(\sqrt{k})}$ time algorithm to compute the optimum solution for bi-DSN_{PLANAR}.

This result is an example of the so-called “square-root phenomenon”: planarity often allows runtimes that improve the exponent by a square root factor in terms of the parameter when compared to the general case [28, 50, 40, 47, 41, 52, 55, 54, 51]. Interestingly though, Chitnis et al. [14] show that under ETH, no $f(k) \cdot n^{o(k)}$ time algorithm can compute the optimum solution to DSN_{PLANAR}. Thus assuming a bidirected input graph in Theorem 4 is necessary (under ETH) to obtain a factor of $O(\sqrt{k})$ in the exponent of $n$. 

esa2018
**Bidirected inputs.** Since in contrast to \( \text{bi-DSN}_{\text{Planar}} \), the bi-DSN problem does not restrict the optimum solutions, one may wonder whether a parameterized approximation scheme as in Theorem 1 is possible for this more general case as well. We answer this in the negative by proving the following result, which implies that restricting the optima to planar graphs was necessary for Theorem 1.

▶ **Theorem 5.** Under Gap-ETH, there exists a constant \( \alpha > 1 \) such that for any computable function \( f(k) \) there is no \( f(k) \cdot n^{O(1)} \) time algorithm that computes an \( \alpha \)-approximation for bi-DSN.

We leave open whether a similar inapproximability result can be obtained for the other obvious generalization of \( \text{bi-DSN}_{\text{Planar}} \), in which the input graph is unrestricted but we need to compute the planar optimum, i.e., the \( \text{DSN}_{\text{Planar}} \) problem. We conjecture that no approximation scheme exists for this problem either.

What approximation factors can be obtained for bi-DSN when parametrizing by \( k \), given the lower bound of Theorem 5 on one hand, and the before-mentioned result [17] that rules out a \( k^{1/4-o(1)} \)-approximation for DSN in time parameterized by \( k \) on the other? It turns out that it is not too hard to obtain a constant approximation for bi-DSN, given the similarity of bidirected graphs to undirected graphs. In particular, relying on the fact that for the undirected version of DSN, i.e. the SF problem, there is a polynomial time 2-approximation algorithm [1], and an FPT algorithm based on [19], we obtain the following theorem, which is also in contrast to Theorem 2.

▶ **Theorem 6.** The bi-DSN problem admits a 4-approximation in polynomial time, and a 2-approximation in \( 2^{O(k \log k)} \cdot n^{O(1)} \) time.

Even if Theorem 5 in particular shows that bi-DSN cannot be FPT under Gap-ETH, it does not give a strong lower bound on the runtime dependence in the exponent of \( n \). However using the weaker ETH assumption we can obtain such a lower bound, as the next theorem shows. Interestingly, the obtained lower bound implies that when aiming for optimum solutions, the restriction to bidirected inputs does not make DSN much easier than the general case, as also for bi-DSN the \( n^{O(k)} \) time algorithm by [24] is essentially best possible. This is in contrast to the bi-DSN\(_{\text{Planar}}\) problem where the square-root phenomenon takes effect as shown by Theorem 4.

▶ **Theorem 7.** The bi-DSN problem is W[1]-hard parameterized by \( k \). Moreover, under ETH there is no \( f(k) \cdot n^{o(k/\log k)} \) time algorithm for bi-DSN, for any computable function \( f(k) \).

Thus when considering bidirected inputs, which lie between directed and undirected graphs, by Theorem 6 the complexity of the bi-DSN problem rather resembles the undirected variant (the SF problem) in terms of approximations, while by Theorem 7 it resembles the directed version (the DSN problem) in terms of FPT algorithms.

**Strongly connected solutions.** Just like the more general DSN problem, the SCSS problem is W[1]-hard [31] parameterized by \( k \), and is also hard to approximate as no polynomial time \( O((\log 2^{-\varepsilon} n)) \)-approximation is possible [32], unless \( \text{NP} \subseteq \text{ZTIME}(n^{\text{polylog}(n)}) \). However it is possible to exploit the structure of the optimum to SCSS to obtain a 2-approximation algorithm parameterized by \( k \), as observed by Chitnis et al. [13]. This is because any strongly connected graph is the union of two arborescences, and these form solutions to DST. The 2-approximation follows, since DST is FPT by the classic result of [19]. Thus in contrast to DSN, for SCSS it is possible to beat any approximation factor obtainable in polynomial time when parametrizing by \( k \).
The SCSS problem admits a $2$-approximation in $3^k \cdot n^{O(1)}$ time.

An obvious question now is whether the approximation ratio of this rather simple algorithm can be improved. Interestingly we are able to show that this is not the case. To the best of our knowledge, this is the first example of a W[1]-hard problem having a parameterized approximation algorithm with non-trivial approximation factor (in this case 2), which is also known to be tight!

Under Gap-ETH, for any $\varepsilon > 0$ and any computable function $f(k)$, there is no $f(k) \cdot n^{O(1)}$ time algorithm that computes a $(2 - \varepsilon)$-approximation for SCSS.

Bidirected inputs with strongly connected solutions. In light of the above results for restricted cases of DSN, what can be said about restricted cases of SCSS? It is implicit in the work of Chitnis et al. [14] that SCSS_{PLANAR}, i.e., the problem of computing the optimum strongly connected planar optimum, can be solved in $2^{O(k \log k)} \cdot n^{O(\sqrt{k})}$ time, while under ETH no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm is possible. Hence SCSS_{PLANAR} is slightly easier than DSN_{PLANAR} where the exponent of $n$ needs to be linear in $k$, as mentioned before. On the other hand, the bi-SCSS problem turns out to be a lot easier to solve than bi-DSN. This is implied by the next theorem, which stands in contrast to Theorem 5 and Theorem 7. In particular, the in-between bi-SCSS problem behaves more like the undirected ST problem than the directed SCSS problem.

There is a $2^{O(2^{k^2 - k})} \cdot n^{O(1)}$ time algorithm for bi-SCSS, i.e., it is FPT for parameter $k$.

Could it be that bi-SCSS is even solvable in polynomial time? We prove that this is not the case, as it is NP-hard. To the best of our knowledge, the class of bidirected graphs is the first example where SCSS remains NP-hard but turns out to be FPT parameterized by $k$! Moreover, note that the above algorithm has a doubly exponential runtime in $k^2$. We conjecture that a single exponential runtime should suffice, and we also obtain a lower bound result of this form, even if we restrict the optimum solutions to very simple planar graphs, namely cycles.

The bi-SCSS_{CYCLE} problem is NP-hard. Moreover, under ETH there is no $2^{o(k)} \cdot n^{O(1)}$ time algorithm for bi-SCSS_{CYCLE}.

Remark. For ease of notation, throughout this paper we chose to use the number of demands $k$ uniformly as the parameter. Alternatively one might also consider the smaller parameter $|R|$, where $R = \bigcup_{i=1}^k \{s_i, t_i\}$ is the set of terminals. Note for instance that in case of the SCSS problem, $k = |R|$, while for DSN, $k$ can be as large as $\Theta(|R|^2)$ (cf. [22]). However we always have $k \geq |R|/2$, since the demands can form a matching in the worst case. It is interesting to note that all our algorithms for DSN have the same running time for parameter $|R|$ as for parameter $k$. That is, we may set $k = |R|$ in Theorem 1, 4, and 6.

1.2 Our techniques

It is already apparent from the above exposition of our results, that understanding the structure of the optimum solution is a powerful tool when studying DSN and its related problems (see Table 1). This is also apparent when reading the literature on these problems, and we draw some of our inspiration from these known results, as described below.
Parameterized Approximations for Bidirected Steiner Networks

Table 1 Summary of achievable runtimes for DSN and SCSS when parameterizing by $k$. Some of the previous results are implicit and, in the papers, are rather stated for the case when the input graphs are restricted to the same class as the optimum solutions. Non-bracketed reference numbers refer to theorems of this paper.

<table>
<thead>
<tr>
<th>problem</th>
<th>approx.</th>
<th>algorithms runtime</th>
<th>ref.</th>
<th>lower bounds approx.</th>
<th>runtime</th>
<th>ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSN</td>
<td>–</td>
<td>$n^{O(k)}$</td>
<td>[24]</td>
<td>–</td>
<td>$f(k)$  $n^{O(k)}$</td>
<td>[31]</td>
</tr>
<tr>
<td>DSN</td>
<td>$O(k^{1+\varepsilon})$</td>
<td>$n^{O(1)}$</td>
<td>[9]</td>
<td>$k^{\frac{1}{2}+o(1)}$</td>
<td>$f(k)$  $n^{O(1)}$</td>
<td>[17]</td>
</tr>
<tr>
<td>DSNTW $\cdot \omega$</td>
<td>–</td>
<td>$2^{O(k\omega \log \omega)} \cdot n^{O(\omega)}$</td>
<td>[27]</td>
<td>–</td>
<td>$f(k, \omega)$  $n^{O(\omega)}$</td>
<td>[27]</td>
</tr>
<tr>
<td>bi-DSN$_{PLANAR}$</td>
<td>$1+\varepsilon$</td>
<td>$\max {2^{k^{O(1)/\varepsilon}}, n^{2^{O(1)/\varepsilon}}}$</td>
<td>1</td>
<td>$1+\varepsilon$</td>
<td>$f(\varepsilon, k)$  $n^{O(\sqrt{\varepsilon})}$</td>
<td>3</td>
</tr>
<tr>
<td>bi-DSN$_{PLANAR}$</td>
<td>–</td>
<td>$2^{O(k^{3/2} \log k)} \cdot n^{O(\sqrt{\varepsilon})}$</td>
<td>4</td>
<td>–</td>
<td>$f(k)$  $n^{O(\sqrt{\varepsilon})}$</td>
<td>3</td>
</tr>
<tr>
<td>DSN$_{PLANAR}$</td>
<td>–</td>
<td>$n^{O(k)}$</td>
<td>[24]</td>
<td>–</td>
<td>$f(k)$  $n^{O(k)}$</td>
<td>[14]</td>
</tr>
<tr>
<td>bi-DSN</td>
<td>–</td>
<td>$n^{O(k)}$</td>
<td>[24]</td>
<td>–</td>
<td>$f(k)$  $n^{O(k)}$</td>
<td>[14]</td>
</tr>
<tr>
<td>bi-DSN</td>
<td>2</td>
<td>$2^{O(k \log k)} \cdot n^{O(1)}$</td>
<td>6</td>
<td>$\alpha \in O(1)$</td>
<td>$f(k)$  $n^{O(1)}$</td>
<td>5</td>
</tr>
<tr>
<td>bi-DSN</td>
<td>4</td>
<td>$n^{O(1)}$</td>
<td>6</td>
<td>$\alpha \in O(1)$</td>
<td>$n^{O(1)}$</td>
<td>2</td>
</tr>
<tr>
<td>SCSS</td>
<td>–</td>
<td>$n^{O(k)}$</td>
<td>[24]</td>
<td>–</td>
<td>$f(k)$  $n^{O(k/\log k)}$</td>
<td>[14]</td>
</tr>
<tr>
<td>SCSS$_{PLANAR}$</td>
<td>2</td>
<td>$3^{k} \cdot n^{O(1)}$</td>
<td>[13]</td>
<td>$2-\varepsilon$</td>
<td>$f(k)$  $n^{O(1)}$</td>
<td>9</td>
</tr>
<tr>
<td>bi-SCSS</td>
<td>–</td>
<td>$2^{O(2^{k/\varepsilon})} \cdot n^{O(1)}$</td>
<td>[14]</td>
<td>–</td>
<td>$2^{\varepsilon} \cdot n^{O(1)}$</td>
<td>11</td>
</tr>
</tbody>
</table>

For our approximation scheme for bi-DSN$_{PLANAR}$, we generalize the insights on the structure of optimum solutions to the classical Steiner Tree (ST) problem for our main result in Theorem 1. For the ST problem, an undirected edge-weighted graph is given together with a terminal set $R$, and the task is to compute the cheapest tree connecting all $k$ terminals. For the ST problem only polynomial time 2-approximations were known [30, 60], until it was taken into account [36, 56, 63, 58] that any optimum Steiner tree can be decomposed into so-called full components, i.e., subtrees for which exactly the leaves are terminals. If a full component contains only a small subset of size $k'$ of the terminals, it is the solution to an ST instance, for which the optimum can be computed efficiently in time $(2+\delta)^{k'} \cdot n^{O(1)}$ for any constant $\delta > 0$ using the algorithm of Mölle et al. [53]. A fundamental observation proved by Borchers and Du [6] is that for any $k'$ there exists a solution to ST of cost at most $1 + \frac{1}{\log_2 k'}$ times the optimum, in which every full component contains at most $k'$ terminals. Thus setting $k' = 2^{1/\varepsilon}$ for some constant $\varepsilon > 0$, all full-components with at most $2^{1/\varepsilon}$ terminals can be computed in polynomial time, and among them exists a collection forming a $(1+\varepsilon)$-approximation. The key to obtain approximation ratios smaller than 2 for ST is to cleverly select a good subset of all computed full-components. This is for instance done in [7] via an iterative rounding procedure, resulting in an approximation ratio of $\ln(4) + \varepsilon < 1.39$, which currently is the best one known.

Our main technical contribution is to generalize the Borchers-Du [6] Theorem to the bi-DSN$_{PLANAR}$ problem. In particular, to obtain our approximation scheme of Theorem 1, we employ a similar approach by decomposing a bi-DSN$_{PLANAR}$ solution into sub-instances, each containing a small number of terminals. As bi-DSN$_{PLANAR}$ is $W[1]$-hard by Theorem 3, we cannot hope to compute optimum solutions to each sub-instance as efficiently as for ST.
However, we provide an XP-algorithm with runtime \(2^{O(k^{3/2} \log k)} \cdot n^{O(\sqrt{k})}\) for \(\text{bi-DSN}_{\text{PLANAR}}\) in Theorem 4. Thus if every sub-instance contains at most \(2^{1/\varepsilon}\) terminals, each can be solved in \(n^{2^{O(1/\varepsilon)}}\) time, and this accounts for the “non-efficient” runtime of our approximation scheme. Since we allow runtimes parameterized by \(k\), we can then exhaustively search for a good subset of precomputed small optimum solutions to obtain a solution to the given demand set \(D\). For the latter solution to be a \((1 + \varepsilon)\)-approximation however, we need to generalize the Borchers-Du [6] Theorem for ST to \(\text{bi-DSN}_{\text{PLANAR}}\) (see Theorem 13 for the formal statement). This constitutes the bulk of the work to prove Theorem 1.

For our exact algorithms for \(\text{bi-DSN}_{\text{PLANAR}}\) and \(\text{bi-SCSS}\), we note that also from a parameterized point of view, understanding the structure of the optimum solution to DSN has lead to useful insights in the past. We will leverage one such recent result by Feldmann and Marx [27]. In [27] the above mentioned standard special case of restricting the patterns of the demands in \(D\) is studied in depth. The result is a complete dichotomy over which classes of restricted patterns define special cases of DSN that are FPT and which are \(W[1]\)-hard for parameter \(k\). The high-level idea is that whenever the demand patterns imply optimum solutions of constant treewidth, there is an FPT algorithm computing such an optimum. In contrast, the problem is \(W[1]\)-hard whenever the demand patterns imply the existence of optimum solutions of arbitrarily large treewidth. The FPT algorithm from [27] lies at the heart of all our positive results, and therefore shows that the techniques developed in [27] to optimally solve special cases of DSN can be extended to find (near-)optimum solutions for other \(W[1]\)-hard special cases as well. It is important to note that the algorithm of [27] can also be used to compute the cheapest solution of treewidth at most \(\omega\), even if there is an even better solution of treewidth larger than \(\omega\) (which might be hard to compute). Formally, the result leveraged in this paper is the following.

> **Theorem 12** (implicit in Theorem 5 of [27]). If \(\mathcal{K}\) is the class of graphs with treewidth at most \(\omega\), then the DSN\(_{\mathcal{K}}\) problem can be solved in time \(2^{O(k \omega \log \omega)} \cdot n^{O(\omega)}\).

We exploit the algorithm given in Theorem 12 to prove our algorithmic results of Theorem 4 and Theorem 10. In particular, we prove that any \(\text{bi-DSN}_{\text{PLANAR}}\) solution has treewidth \(O(\sqrt{k})\), from which Theorem 4 follows immediately. For \(\text{bi-SCSS}\) however, we give an example of an optimum solution of treewidth \(\Omega(k)\). Hence we cannot exploit the algorithm of Theorem 12 directly to obtain Theorem 10. In fact on general input graphs, a treewidth of \(\Omega(k)\) would imply that the problem is \(W[1]\)-hard by the hardness results in [27] (which was indeed originally shown by Guo et al. [31]). As this stands in stark contrast to Theorem 10, it is particularly interesting that the problem on bidirected input graphs is FPT. We prove this result by decomposing an optimum solution to \(\text{bi-SCSS}\) into instances of \(\text{bi-SCSS}_{\mathcal{K}}\), where \(\mathcal{K}\) is the class of directed graphs of treewidth 1 (so-called poly-trees). For each such sub-instance we can compute a solution in \(2^{O(k)} \cdot n^{O(1)}\) time by using Theorem 12 (for \(\omega = 1\), and then combine them into an optimum solution to \(\text{bi-SCSS}\).

Our hardness proofs for \(\text{bi-DSN}\) are based on reductions from the Grid Tiling problem [15]. This problem is particularly suited to prove hardness for problems on planar graphs, due to its grid-like structure. We first develop a specific gadget that can be exploited to show hardness for bidirected graphs. This gadget however is not planar. We only exploit the structure of Grid Tiling to show that the optimum solution is planar for Theorem 3. For Theorem 7 we modify this reduction to obtain a stronger runtime lower bound, but in the process we lose the property that the optimum is planar.

Our parameterized inapproximability result for SCSS is proved by combining a variant of a known reduction by Guo et al. [31] with a recent parameterized hardness of approximation
An approximation scheme for bi-DSN$_{\text{Planar}}$

In this section we prove Theorem 1. Note that since we have $k$ demand pairs, it follows that the number of terminals $|R|$ is at most $2k$, where $R = \bigcup_{i=1}^{k} \{s_i, t_i\}$. Henceforth in this section, we use the upper bound $2k$ on the number of terminals $|R|$ for ease of presentation (when instead we could replace $k$ by $|R|$ in the running time of Theorem 1). The bulk of the proof is captured by the following result, which generalizes the corresponding theorem by Borchers and Du [6] for the ST problem, and which is our main technical contribution. In order to facilitate the definition of a sub-instance to DSN, we encode the demands of a DSN instance using a pattern graph $H$, as also done in [27]: the vertex set of $H$ is the terminal set $R$, and $H$ contains the directed edge $st$ if and only if $(s,t)$ is a demand. Hence the DSN problem asks for a minimum cost network $N \subseteq G$ having an $s \rightarrow t$ path for each edge $st$ of $H$.

**Theorem 13.** Let $G$ be a bidirected graph, and $H$ a pattern graph on $R \subseteq V(G)$. Let $N \subseteq G$ be an optimum bi-DSN$_{\text{Planar}}$ solution to $H$, i.e. $N$ is planar. For any $\varepsilon > 0$, there exists a set of patterns $\mathcal{H}$ such that for each $H' \in \mathcal{H}$ there is a feasible bi-DSN$_{\text{Planar}}$ solution $N_{H'} \subseteq G$ and $|V(H')| \leq O(1/\varepsilon)$. Furthermore, the union $\bigcup_{H' \in \mathcal{H}} N_{H'}$ of these solutions forms a feasible bi-DSN$_{\text{Planar}}$ solution to $H$ with $\sum_{H' \in \mathcal{H}} \text{cost}(N_{H'}) \leq (1 + \varepsilon) \cdot \text{cost}(N)$.

Based on Theorem 13 our $(1 + \varepsilon)$-approximation algorithm proceeds as follows. The first step is to compute an optimum solution for every possible pattern graph on at most $g(\varepsilon) = 2^{O(1/\varepsilon)}$ terminals. Since any pattern graph has at most $2^{O(1/\varepsilon)} < g(\varepsilon)^2$ edges, and there is a total of $2^{g(\varepsilon)/2} < 8k^2$ possible demands between the $2k$ terminals, the total number of pattern graphs is $O(k^{2g(\varepsilon)^2}) = k^{2^{O(1/\varepsilon)}}$. For each pattern $G$ the algorithm computes the optimum bi-DSN$_{\text{Planar}}$ solution in time $2^{O(k^{2g(\varepsilon)^2})} \cdot n^{O(\sqrt{g(\varepsilon)})} = n^{2^{O(1/\varepsilon)}}$ using the algorithm of Theorem 4. This amounts to a total runtime of $k^{2^{O(1/\varepsilon)}} \cdot n^{2^{O(1/\varepsilon)}}$ up to this point. The algorithm then proceeds by considering each subset $H$ of the pattern graphs, and checking whether the union of the precomputed optimum solutions to all $H' \in H$ forms a feasible solution to the input pattern $H$ on $R$. As there are $2^{O(k^{2g(\varepsilon)^2})}$ subsets $H$, and checking whether a subset induces a feasible solution can be done in polynomial time, this takes $2^{O(k^{2g(\varepsilon)^2})} \cdot n^{O(1)} = 2^{2^{O(1/\varepsilon)}} \cdot n^{O(1)}$ time. Among all feasible unions the algorithm outputs the solution with smallest cost. According to Theorem 13 this solution is a $(1 + \varepsilon)$-approximation, and the total runtime is $k^{2^{O(1/\varepsilon)}} \cdot n^{2^{O(1/\varepsilon)}} + 2^{2^{O(1/\varepsilon)}} \cdot n^{O(1)} = \max \left\{ 2^{2^{O(1/\varepsilon)}}, n^{2^{O(1/\varepsilon)}} \right\}$. Thus we obtain Theorem 1.

Note that even though the output of the algorithm is a $(1 + \varepsilon)$-approximation to the optimum bi-DSN$_{\text{Planar}}$ solution, the computed solution may not be planar, as it is the union of several planar graphs. Theorem 13 shows though that the structure of the optimum can be exploited to compute a near-optimum solution. We also note that the Borchers-Du[6] Theorem for the ST problem implies the existence of a polynomial-sized $(1 + \varepsilon)$-approximate kernel for ST, as recently shown by Lokshakov et al. [46]. By the same arguments this is also true for bi-DSN$_{\text{Planar}}$, due to Theorem 13. We refer to [46] for more details.

**Corollary 14 (cf. [46]).** The bi-DSN$_{\text{Planar}}$ problem admits a polynomial-size approximate kernelization scheme (PSAKS) parameterized by $k$. 

It remains to prove Theorem 13. For this we assume we know the optimum planar solution \( N \subseteq G \), and first use a standard transformation on \( N \), so that each terminal has only 1 neighbour, each Steiner vertex has exactly 3 neighbours, and every pair of edges \( uv \) and \( vu \) have unique costs. Furthermore, let \( G_N \) be the graph spanned by the edge set \( \{uv, vu \in E(G) \mid u \in E(N)\} \), i.e. it is the underlying bidirected graph of \( N \) after performing the transformations on \( N \). In particular, also in \( G_N \) each terminal has only 1 neighbour, each Steiner vertex has exactly 3 neighbours, and every pair of edges \( uv \) and \( vu \) have unique costs. It is not hard to see that proving Theorem 13 for the obtained optimum solution \( N \) in \( G_N \) implies the same result for the original optimum solution in \( G \), by reversing all transformations.

The proof consists of two parts, of which the first exploits the bidirectedness of \( G_N \), while the second exploits that the optimum \( N \) is planar. The first part will identify paths connecting each Steiner vertex to some terminal in such a way that the paths do not overlap much. This will enable us to select a subset of these paths in the second part, so that the total weight of the selected paths is an \( \varepsilon \)-fraction of the cost of the optimum solution. This subset of paths will be used to connect terminals to the boundary vertices of small regions into which we divide the optimum. These regions extended by the paths then form solutions to sub-instances to DSN, which together have a cost of \( 1 + \varepsilon \) times the optimum. The first part is captured by the next lemma, where \( \text{cost}(G') \) denotes the total edge weight of a graph \( G' \).

\[ \text{Lemma 15.} \text{ Let } N \subseteq G_N \text{ be the optimum bi-DSN}^{\text{PLANAR}} \text{ solution to a pattern graph } H \text{ on } R \subseteq V(G_N). \text{ For every } \text{Steiner vertex } v \in V(N) \setminus R \text{ of } N \text{ there is a path } P_v \text{ in } G_N, \text{ such that } P_v \text{ is a } v \rightarrow t \text{ path to some terminal } t \in R, \text{ and the total cost } \sum_{v \in V(N) \setminus R} \text{cost}(P_v) \text{ of these paths is } O(\text{cost}(N)). \]

For the second part we give each vertex \( v \) of \( N \) a weight \( c(v) \), which is zero for terminals and equal to \( \text{cost}(P_v) \) for each Steiner vertex \( v \in V(N) \setminus R \) and corresponding path \( P_v \) given by Lemma 15. We now divide the optimum solution \( N \) into regions of small size, such that the boundaries of the regions have small total weight. Formally, a region is a subgraph of \( N \), and an \( r \)-division is given by a partition of the edges of \( N \), each spanning a region with at most \( r \) vertices. A boundary vertex of an \( r \)-division is a vertex that lies in at least two regions. In a weak \( r \)-division, as for instance defined in [35], we bound the total number of boundary vertices and the number of regions (it is called “weak” since it does not bound the boundary vertices of each region individually). For unweighted planar graphs it can be shown that there is an \( r \)-division with only \( O(n/\sqrt{r}) \) boundary vertices and \( O(n/r) \) regions [35, 29]. To prove this, a separator theorem is applied recursively until each resulting region is small enough. The bound on the number of boundary vertices follows from the well-known fact that any planar graph has a small separator of size \( O(\sqrt{n}) \).

We however need to bound the total weight of the boundary vertices, i.e. we need a weighted weak \( r \)-division. Unfortunately, separator theorems are not helpful here, since they only bound the number of vertices in the separator but cannot bound their weight. Instead we leverage techniques developed for the Klein-Plotkin-Rao (KPR) Theorem \([45, 39]\) in order to show that there is an \( r \)-division for which the total weight of all boundary vertices is an \( O(1/\log r) \)-fraction of the total weight \( \sum_{v \in V(N)} c(v) \), if the graph has constant maximum degree. We later set \( r = 2^{1/\varepsilon} \) in order to obtain an \( \varepsilon \)-fraction of the total weight. Even though the obtained fraction is exponentially worse than the \( O(1/\sqrt{r}) \)-fraction for unweighted graphs obtained in [35, 29], it follows from a lower bound result of Borchers and Du [6] that for weighted graphs this is best possible, even if the graph is a tree. In contrast to the
unweighted case, we also do not guarantee any bound on the number of regions, and we do not need such a bound either. Our proof follows the outlines of the proof given by Lee [45] for the KPR Theorem. In the following, \( c(S) = \sum_{v \in S} c(v) \) for any set of vertices \( S \).

**Lemma 16.** Let \( N \) be a directed planar graph for which each vertex has at most 3 neighbours, and let each vertex \( v \) of \( N \) have a weight \( c(v) \in \mathbb{R} \). For any \( r \in \mathbb{N} \) there is a partition \( \mathcal{E} \) of the edges of \( N \) for which every set in \( \mathcal{E} \) spans at most \( r \) vertices, and if \( B \) is the set of boundary vertices of the regions spanned by the sets in \( \mathcal{E} \), then \( c(B) = O\left( \frac{c(V(N))}{\log r} \right) \).

We here only prove some parts of Lemma 15 (cf. [12] for the full version of the paper).

**Proof of Lemma 15.** We begin by analysing the structure of optimal DSN solutions in bidirected graphs. Here a *condensation graph* of a directed graph results from contracting each strongly connected component, which hence is a DAG. A *poly-forest* is obtained by directing the edges of an undirected forest.

**Claim 17.** For any solution \( N \subseteq G_N \) to a pattern \( H \), there is a solution \( M \subseteq G_N \) to \( H \) with \( \text{cost}(M) \leq \text{cost}(N) \), such that the condensation graph of \( M \) is a poly-forest.

By Claim 17 we may assume w.l.o.g. that the condensation graph of the optimum solution \( N \) is a poly-forest. Consider a weakly connected component \( C \) of \( N \), i.e. inducing a connected component of the underlying undirected graph \( \tilde{N} \). We first extend \( C \) to a strongly connected graph \( C' \) as follows. Let \( F \) be the edges of \( C \) that do not lie in a strongly connected component, i.e. they are the edges of the condensation graph of \( C \). Let \( \tilde{F} = \{ uv \mid vu \in F \} \) be the set containing the reverse edges of \( F \), and let \( C' \) be the strongly connected graph spanned by all edges of \( C \) in addition to the edges in \( \tilde{F} \). Note that adding \( \tilde{F} \) to \( C \) increases the cost by at most a factor of two as \( G_N \) is bidirected, and the number of neighbours of any vertex does not change. We claim that in fact \( C' \) is a *minimal SCSS* solution to the terminal set \( R_C \subseteq R \) contained in \( C \), that is, removing any edge of \( C' \) will disconnect some terminal pair of \( R_C \).

For this, consider any \( s \to t \) path of \( C' \) containing an edge \( e \in \tilde{F} \) for some terminal pair \( s, t \in R_C \). As the edges \( F \) of the condensation graph of \( C \) form a poly-tree, every path from \( s \) to \( t \) in \( C' \) must pass through \( e \). In particular there is no \( s \to t \) path in \( C \), and thus there is no edge \( st \) in the pattern graph \( H \). Or conversely, for any terminal pair \( s, t \in R_C \) for which there is a demand \( st \in E(H) \), no \( s \to t \) path in \( C' \) passes through an edge of \( \tilde{F} \). Thus the set of paths from \( s \) to \( t \) is the same in \( C' \) and \( C \). Since every edge \( e \) of the weakly connected component \( C \) is necessary for some such pair \( s, t \in R_C \) with \( st \in E(H) \), the edge \( e \) is still necessary in \( C' \). Moreover, for any of the added edges \( uv \in \tilde{F} \) the reverse edge \( vu \in F \) was necessary in \( C \) to connect some \( s \in R_C \) to some \( t \in R_C \). As observed above, \( uv \) is necessary to connect \( t \) to \( s \) in \( C' \), since the edges \( F \) of the condensation graph form a poly-tree.

As \( C' \) is a minimal SCSS solution to the terminals \( R_C \) contained within, it is the union of an in-arborescence \( A_{in} \) and out-arborescence \( A_{out} \), both with the same root \( r \in R_C \) and leaf set \( R_C \setminus \{ r \} \), since every terminal only has one neighbour in \( G_N \). A *branching point* of an arborescence \( A \) is a vertex with at least two children in \( A \). We let \( W \subseteq V(C') \) be the set consisting of all terminals \( R_C \) and all branching points of \( A_{in} \) and \( A_{out} \). We will need that any vertex of \( C' \) has a vertex of \( W \) in its close vicinity. That is, if \( \Delta[v] = \{ u \in V(C') \mid u = v \lor uv \in E(C') \lor vu \in E(C') \} \) denotes the inclusive neighbourhood of a vertex \( v \) ignoring directions of edges and \( \Delta^2[v] = \bigcup_{u \in \Delta[v]} \Delta[u] \), we prove the following.

**Claim 18.** For every vertex \( v \) of \( C' \), there is a vertex of \( W \) in \( \Delta^2[v] \).
As the graph $G_N$ is bidirected, for any $v$-$u$ path $P$ in the underlying undirected graph $\overarrow{G}_N$ of $G_N$, there exists a corresponding directed $v \rightarrow u$ path in $G_N$ of the same cost. Therefore, we can ignore the directions of the edges in $C'$ and the arborescences $A_{\text{out}}$ and $A_{\text{in}}$ to identify the paths $P_v$ for Steiner vertices $v$ of $N$. Thus we will only consider paths in the underlying undirected graphs $\overrightarrow{C}'$, $\overrightarrow{A}_{\text{out}}$, and $\overrightarrow{A}_{\text{in}}$ from now on. In particular, we exploit the following observation found in [20] (and also used by [6]) on undirected trees.

**Claim 19 ([20, Lemma 3.2]).** For any undirected tree $T$ we can find a path $P_v \subseteq T$ for every branching point $v$, such that $P_v$ leads from $v$ to some leaf of $T$, and all these paths $P_v$ are pairwise edge-disjoint.

If a Steiner vertex $v$ of $C'$ is a branching point of $A_{\text{out}}$ ($A_{\text{in}}$), we let $P_v$ be the corresponding path in $\overrightarrow{A}_{\text{out}}$ ($\overrightarrow{A}_{\text{in}}$) given by Claim 19 from $v$ to some leaf of $A_{\text{out}}$ ($A_{\text{in}}$), which is a terminal. Note that paths in $\overrightarrow{A}_{\text{in}}$ may overlap with paths in $\overrightarrow{A}_{\text{out}}$. However any edge in the union of all the paths $P_v$ chosen so far is contained in at most two such paths, one for a branching point of $A_{\text{out}}$ and one for a branching point of $A_{\text{in}}$.

It remains to choose a path $P_v$ for every Steiner vertex $v$ that is neither a branching point of $A_{\text{out}}$ nor of $A_{\text{in}}$, i.e. for every vertex not in $W$. By Claim 18 for any such vertex $v \not\in W$ there is a vertex $u \in \Delta[v]$ for which $u \in W$. If $u$ is a terminal, then the path $P_v$ is simply the edge $vu$ if $u \in \Delta[v]$ or the corresponding path $vwu$ for some $w \in \Delta[v]$ otherwise. If $u$ is not a terminal but a branching point of $A_{\text{out}}$ or $A_{\text{in}}$, then we chose a path $P_v$ for $u$ above. In this case, $P_v$ is the path contained in the walk given by extending the path $P_u$ by the edge $vu$ or the path $vwu$, respectively. Note that, as any vertex of $C'$ has at most three neighbours, any terminal or branching point $u \in W$ can be used in this way for some vertex $v \not\in W$ at most nine times. Therefore any edge in the union of all chosen paths is contained in $O(1)$ paths. Consequently the total cost $\sum_{v \in V(N) \setminus R} \text{cost}(P_v)$ is $O(\text{cost}(C'))$, and as $\text{cost}(C') \leq 2 \text{cost}(C)$ we also get $\sum_{v \in V(N) \setminus R} \text{cost}(P_v) = O(\text{cost}(C))$.

We may repeat these arguments for every weakly connected component of $N$ to obtain the lemma.

**References**

Parameterized Approximations for Bidirected Steiner Networks


Parameterized Approximations for Bidirected Steiner Networks


Near-linear time approximations schemes for clustering in doubling metrics

Vincent Cohen-Addad∗, Andreas Emil Feldmann†, David Saulpic∗‡

∗Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
† Charles University in Prague, Czechia
‡ École Normale Supérieure, Paris, France

Email: vcohenad@gmail.com, feldmann.a.e@gmail.com, david.saulpic@lip6.fr

Abstract—We consider the classic Facility Location, k-Median, and k-Means problems in metric spaces of constant doubling dimension. We give the first nearly linear-time approximation schemes for each problem, making a significant improvement over the state-of-the-art algorithms.

Moreover, we show how to extend the techniques used to get the first efficient approximation schemes for the problems of prize-collecting k-Medians and k-Means, and efficient bicriteria approximation schemes for k-Medians with outliers, k-Means with outliers and k-Center.

I. INTRODUCTION

The k-Median and k-Means problems are classic clustering problems that are highly popular for modeling the problem of computing a “good” partition of a set of points of a metric space into k parts so that points that are “close” should be in the same part. Since a good clustering of a dataset allows to retrieve information from the underlying data, the k-Median and k-Means problems are the cornerstone of various approaches in data analysis and machine learning. The design of efficient algorithms for these clustering problems has thus become an important challenge.

The input for the problems is a set of points in a metric space and the objective is to identify a set of k centers C such that the sum of the pth power of the distance from each point of the metric to its closest center in C is minimized. In the k-Median problem, p is set to 1 while in the k-Means problem, p is set to 2. In general metric spaces both problems are known to be APX-hard, and this hardness even extends to Euclidean spaces of any dimension d = \Omega(\log n) [5]. Both problems also remain NP-hard for points in \mathbb{R}^d [33]. For k-Center, the goal is to minimize the maximum distance from each point in the metric to its closest center. This problem is APX-hard even in Euclidean Spaces [17], and computing a solution with optimal cost but (1 + \varepsilon)k centers requires time at least \Omega(n^{\sqrt{n}/\sqrt{d}}) [30]. Therefore, to get an efficient approximation scheme one needs to approximate both the number of centers and the cost. (See Section II-C for more related work).

To bypass these hardness of approximation results, researchers have considered low-dimensional inputs like Euclidean spaces of fixed dimension or more generally metrics of fixed doubling dimension. There has been a large body of work to design good tools for clustering in metrics of fixed doubling dimension, from the general result of Talwar [37] to very recent coreset constructions for clustering problems [23]. In their seminal work, Arora et al. [4] gave a polynomial time approximation scheme (PTAS) for k-Median in \mathbb{R}^d, which generalizes to a quasi-polynomial time approximation scheme (QPTAS) for inputs in \mathbb{R}^d. This result was improved in two ways. First by Talwar [37] who generalized the result to any metric space of fixed doubling dimension. Second by Kolliopoulos and Rao [25] who obtained an \( f(\varepsilon, d) \cdot n \log^{d/\varepsilon} n \) time algorithm for k-Median in d-dimensional Euclidean space. Unfortunately, Kolliopoulos and Rao’s algorithm relies on the Euclidean structure of the input and does not immediately generalize to low dimensional doubling metric. Thus, until recently the only result known for k-Median in metrics of fixed doubling dimension was a QPTAS. This was also the case for slightly simpler problems such as Uniform Facility Location. Moreover, as pointed out in [12], the classic approach of Arora et al. [4] cannot work for the k-Means problem. Thus no efficient algorithms were known for the k-Means problem, even in the plane.

Recently, Friggstad et al. [19] and Cohen-Addad et al. [13] showed that the classic local search algorithm for the problems gives a (1 + \varepsilon)-approximation in time \( n^{1/(d/c)} \) in Euclidean space, in time \( n^{O(1/c^2)} \) for planar graphs (which also extends to minor-free graphs), and in time \( n^{(d/c)^{\Theta(c)}} \) in metrics of doubling dimension d [19]. More recently Cohen-Addad [12] showed how to speed up the local search algorithm for Euclidean space to obtain a PTAS with running time \( nk(\log n)^{(d/c)^{\Theta(c)}} \).

Nonetheless, obtaining an efficient approximation scheme (namely an algorithm running in time \( f(\varepsilon, d)\text{poly}(n) \)) for k-Median and k-Means in metrics of doubling dimension d has remained a major challenge.

The versatility of the techniques we develop to tackle these problems allows us to consider a broader setting, where the clients do not necessarily have to be served. In the prize-collecting version of the problems, every client has a penalty cost that can be paid instead of its serving cost. In the k-Median (resp. k-Means) with outliers problems, the goal is to serve all but z clients, and the cost is measured on
the remaining ones with the k-Median (resp. k-Means) cost. These objectives can help to handle some noise from the input: the k-Median objective can be dramatically perturbed by the addition of a few distant clients, which must then be discarded.

A. Our Results

We solve this open problem by proposing the first near-linear time algorithms for the k-Median and k-Means problems in metrics of fixed doubling dimension. More precisely, we show the following theorems, where we let $f(\varepsilon) = (1/\varepsilon)^{1/\varepsilon}$.

**Theorem 1.1.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1 + \varepsilon)$-approximation algorithm for k-Median in metrics of doubling dimension $d$ with running time $O(n^{2d+1} \log^4 n + 2^{O(d)} n \log^5 n)$ and success probability at least $1 - \varepsilon$.

**Theorem 1.2.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1 + \varepsilon)$-approximation algorithm for k-Means in metrics of doubling dimension $d$ with running time $f(\varepsilon)^{2^{O(d)} n \log^4 n + 2^{O(d)} n \log^5 n}$ and success probability at least $1 - \varepsilon$.

Our results also extend to the Facility Location problem, in which no bound on the number of opened centers is given, but each center comes with an opening cost. The aim is to minimize the sum of the (1st power) of the distances from each point of the metric to its closest center, in addition to the total opening costs of all used centers.

**Theorem 1.3.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1 + \varepsilon)$-approximation algorithm for Facility Location in metrics of doubling dimension $d$ with running time $O(n^{2d+1} \cdot n + 2^{O(d)} n \log n)$ and success probability at least $1 - \varepsilon$.

In all these theorems, we make the common assumption to have access to the distances of the metric in constant time, as, e.g., in [22], [15], [20]. This assumption is discussed in Bartal et al. [7].

Note that the double-exponential dependence on $d$ is unavoidable unless P = NP, since the problems are APX-hard in Euclidean space of dimension $d = O(\log n)$. For Euclidean inputs, our algorithms for the k-Means and k-Median problems outperform the ones of Cohen-Addad [12], removing in particular the dependence on $k$, and the one of Kolliopoulos and Rao [25] when $d > 3$, by removing the dependence on $\log^d n$. Interestingly, for $k = \omega(\log^a n)$ our algorithm for the k-Means problem is faster than popular heuristics like k-Means++ which runs in time $O(n k)$ in Euclidean space.

We note that the success probability can be boosted to $1 - \varepsilon^2$ by repeating the algorithm log $\delta$ times and outputting the best solution encountered.

After proving the three theorems above, we will apply the techniques to prove the following ones. We say an algorithm is an $(\alpha, \beta)$-approximation for k-Medians or k-Means with outliers if its cost is within an $\alpha$ factor of the optimal one and the solution drops $\beta$ outliers. Similarly, an algorithm is an $(\alpha, \beta)$-approximation for k-Center if its cost is within an $\alpha$ factor of the optimal one and the solution opens $\beta k$ centers.

**Theorem 1.4.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1 + \varepsilon)$-approximation algorithm for Prize-Collecting k-Median (resp. k-Means) in metrics of doubling dimension $d$ with running time $O(n^{2d+1} \log^4 n + 2^{O(d)} n \log^5 n)$ and success probability at least $1 - \varepsilon$.

**Theorem 1.5.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1 + \varepsilon, 1 + O(\varepsilon))$-approximation algorithm for k-Median (resp. k-Means) with outliers in metrics of doubling dimension $d$ with running time $O(n^{2d+1} \log^4 n + T(n))$ and success probability at least $1 - \varepsilon$, where $T(n)$ is the running time to construct a constant-factor approximation.

We note as an aside that our proof of Theorem 1.5 could give an approximation where at most $\varepsilon$ outliers are dropped, but $(1 + O(\varepsilon))k$ centers are opened. For simplicity, we focused on the previous case.

**Theorem 1.6.** For any $0 < \varepsilon < 1/3$, there exists a randomized $(1 + \varepsilon, 1 + O(\varepsilon))$-approximation algorithm for k-Center in metrics of doubling dimension $d$, with running time $O(n^{2d+1} \log^4 n + \log n)$ and success probability at least $1 - \varepsilon$.

As explained above, this bicriteria is necessary in order to get an efficient algorithm: it is APX-hard to approximate the cost [17], and achieving the optimal cost with $(1 + \varepsilon)k$ centers requires a complexity $\Omega(n^{1/\sqrt{\varepsilon}})$ [30].

B. Techniques

To give a detailed insight on our techniques and our contribution we first need to quickly review previous approaches for obtaining approximation schemes on bounded doubling metrics. The general approach, due to Arora [3] and Mitchell [36], which was generalized to doubling metrics by Talwar [37], is the following.

1) Previous Techniques: The approach consists in randomly partitioning the metric into a constant number of regions, and applying this recursively to each region. The recursion stops when the regions contain only a constant number of input points. This leads to what is called a split-tree decomposition: a partition of the space into a finer and finer level of granularity. The reader who is not familiar with the split-tree decomposition of Talwar may refer to Section II-B for a more formal introduction.

Portals: The approach then identifies a specific set of points for each region, called portals, which allows to show
that there exists a near-optimal solution such that different regions “interplay” only through portals. For example, in the case of the Traveling Salesperson (TSP) problem, it is possible to show that there exists a near-optimal tour that enters and leaves a region only through its portals. In the case of the \( k \)-Median problem a client located in a specific region can be assigned to a facility in a different region only through a path that goes to a portal of the region. In other words, clients can “leave” a region only through the portals.

Proving the existence of such a structured near-optimal solution relies on the fact that the probability that two very close points end up in different regions of large diameter is very unlikely. Hence the expected detour paid by going through a portal of the region is small compared to the original distance between the two points, if the portals are dense enough.

For the sake of argument, we provide a proof sketch of the standard proof of Arora [3]. We will use a refined version of this idea in later sections. The split-tree recursively divides the input metric \((V, \text{dist})\) into parts of smaller and smaller diameter. The root part consists of the entire point set and the parts at level \(i\) are of diameter roughly \(2^i\). The set of portals of a part of level \(i\) is an \(\varepsilon_i \cdot 2^i\)-net for some \(\varepsilon_0\), which is a small set such that every point of the metric is at distance at most \(\varepsilon_i 2^i\) to it. Consider two points \(u, v\) and let us bound the expected detour incurred by connecting \(u\) to \(v\) through portals. This detour is determined by a path that starting from \(u\) at the lowest level, in each step connects a vertex \(w\) to its closest net point of the part containing \(w\) on the next higher level. This is done until the lowest-level part \(R_{u,v}\) (i.e., the part of smallest diameter) is reached, which contains both \(u\) and \(v\), from where a similar procedure leads from this level through portals of smaller and smaller levels all the way down to \(v\). If the level of \(R_{u,v}\) is \(i\) then the detour, i.e., the difference between \(\text{dist}(u, v)\) and the length of the path connecting \(u\) and \(v\) through portals, is \(O(\varepsilon_i 2^i)\) by the definition of the net. Moreover, the proof shows that the probability that \(u\) and \(v\) are not in the same part on level \(i\) is at most \(\varepsilon_i^2 \cdot 2^i\). Thus, the expected detour for connecting \(u\) to \(v\) is \(\sum_{i=0}^{\infty} \text{Pr}[R_{u,v}\text{ is at level } i] \cdot O(\varepsilon_i 2^i) = \sum_{i=0}^{\infty} \varepsilon_i 2^i \text{dist}(u, v)\). Hence, setting \(\varepsilon_0\) to be some \(\varepsilon\) divided by the number of levels yields that the expected detour is \(O(\varepsilon \text{dist}(u, v))\).

**Dynamic programming:** The portals now act as separators between different parts and allows to apply a dynamic programming (DP) approach for solving the problems. The DP consists of a DP-table entry for each pair and for each configuration of the portals of the part. Here a configuration is a potential way the near-optimal solution interacts with the part. For example, in the case of TSP, a configuration is the information at which portal the near-optimal tour enters and leaves and how it connects the portals on the outside and inside of the part. For the \(k\)-Median problem, a configuration stores how many clients outside (respectively inside) the part connect through each portal and are served by a center located inside (respectively outside). Then the dynamic program proceeds in a bottom-up fashion along the split-tree to fill up the DP table. The running time of the dynamic program depends exponentially on the number of portals.

**How many portals?** The challenges that need to be overcome when applying this approach, and in particular to clustering problems, are two-fold. First the “standard” use of the split-tree requires \(O((\log n)^d)\) portals per part in order to obtain a \((1 + \varepsilon)\)-approximation, coming from the fact that the number of levels can be assumed to be logarithmic in the number of input points. This often implies quasi-polynomial time approximation schemes since the running time of the dynamic program has exponential dependence on the number of portals. This is indeed the case in the original paper by Talwar [37] and in the first result on clustering in Euclidean space by Arora et al. [4]. However, in some cases, one can lower the number of portals per part needed. In Euclidean space for example, the celebrated “patching lemma” [2] shows that only a constant number (depending on \(\varepsilon\)) of portals are needed for TSP. Similarly, Kalliopoulos and Rao [25] showed that for \(k\)-Median in Euclidean space only a constant number of portal are needed, if one uses a slightly different decomposition of the metric.

Surprisingly, obtaining such a result for doubling metrics is much more challenging. To the best of our knowledge, this work is the first one to reduce the number of portals to a constant.

A second challenge when working with split-tree decompositions and the \(k\)-Means problem is that because the cost of assigning a point to a center is the squared distance, the analysis of Arora, Mitchell, and Talwar does not apply. If two points are separated at a high level of the split-tree, then making a detour to the closest portal may incur an expected cost much higher than the cost of the optimal solution.

2) **Our Contributions:** Our contribution can be viewed as a “patching lemma” for clustering problems in doubling metrics. Namely, an approach that allows to solve the problems mentioned above: (1) it shows how to reduce the number of portals to a constant, similar to the one given by the patching lemma for TSP, (2) it works for any clustering objective which is defined as the sum of distances to some constant \(p\) (with \(k\)-Median and \(k\)-Means as prominent special cases), and (3) it works not only for Euclidean but also for doubling metrics.

Our starting point is the notion of *badly cut* vertices of Cohen-Addad [11] for the capacitated version of the above clustering problems. To provide some intuition on the definition, let us start with the following observation: consider a center \(f\) of the optimal solution and a client \(c\) assigned to \(f\). If the diameter of the lowest-level part containing both \(f\) and \(c\) is of order \(\text{dist}(c, f)\) (say at most

---

542
dist(c, f)/\varepsilon^2\), then by taking a large enough but constant size net as a set of portals in each part (say an \(\varepsilon^2\)-net for a part of level \(i\)), the total detour for the two points is at most \(O(\varepsilon \text{dist}(c, f))\), which is acceptable.

The problematic scenario is when the lowest-level part containing \(f\) and \(c\) is of diameter much larger than \(\text{dist}(c, f)\). In this case, it is impossible to afford a detour proportional to the diameter of the part in the case of the \(k\)-Medians and \(k\)-Means objective. To handle this case we first compute a constant approximation \(L\) (via some known algorithm) and use it to guide us towards a \((1 + \varepsilon)\)-approximation.

**Badly cut clients and facilities:** Consider a client \(c\) and the center \(L(c)\) serving \(c\) in \(L\) (i.e., \(L(c)\) is closest to \(c\) among the centers in \(L\)), and call OPT\((c)\) the facility of an optimum solution OPT that serves \(c\) in OPT. We say that \(c\) is badly cut if there is a point \(q\) in the ball centered at \(c\) of radius \(\varepsilon \text{dist}(c, L(c))/\varepsilon\) such that the highest-level part containing \(c\) and not \(q\) is of diameter much larger than \(\text{dist}(c, L(c))/\varepsilon\) (say greater than \(\text{dist}(c, L(c))/\varepsilon^3\)). In other words, there is a point \(q\) in this ball such that paying a detour through the portal to connect \(c\) to \(q\) yields a detour larger than \(\varepsilon \text{dist}(c, q)\) (see Fig. 1).

Similarly, we say that a center \(l\) is badly cut if there is a point \(q\) in the ball centered at \(l\) of radius \(\varepsilon \text{dist}(l, f_0)/\varepsilon^2\) (where \(f_0\) is the facility of OPT that is the closest to \(l\)) such that the highest-level part containing \(l\) and not \(q\) is of diameter \(\varepsilon \text{dist}(l, f_0)/\varepsilon^2\). The crucial property here is that any client \(c\) or any facility \(l\) is badly cut with probability \(O(\varepsilon^3)\), as we will show.

**Using the notion of badly cut:** We now illustrate how this notion can help us. Assume for simplicity that OPT\((c)\) is in the ball centered at a client \(c\) of radius \(\varepsilon \text{dist}(c, L(c))/\varepsilon\) (if this is not the case then \(\text{dist}(c, \text{OPT}(c))\) is much larger than \(\varepsilon \text{dist}(c, L(c))\), so this is a less problematic scenario and a simple idea can handle it). If \(c\) is not badly cut, then the lowest-level part containing both \(c\) and OPT\((c)\) is of diameter not much larger than \(\varepsilon \text{dist}(c, L(c))/\varepsilon\). Taking a sufficiently fine net for each part (independent of the number of levels) allows to bound the detour through the portals to reach \(\text{OPT}(c)\) from \(c\) by at most \(\varepsilon \text{dist}(c, L(c))\). Since \(L\) is an \(O(1)\)-approximation, this is fine.

If \(c\) is badly cut, then we modify the instance by relocating \(c\) to \(L(c)\). That is, we will work with the instance where there is no more client at \(c\) and there is an additional client at \(L(c)\). We claim that any solution in the modified instance can be lifted to the original instance at an expected additional cost of \(O(\varepsilon^3 \text{OPT})\). This comes from the fact that the cost increase for a solution is, by the triangle inequality, at most the sum of distances of the badly cut clients to their closest facility in the local solution. This is at most \(O(\varepsilon^3 \text{OPT})\) in expectation since each client is badly cut with probability at most \(O(\varepsilon^3)\) and \(L\) is an \(O(1)\)-approximation.

Here we should ask, what did we achieve by moving \(c\) to \(L(c)\)? Note that \(c\) should now be assigned to facility \(f\) of OPT that is the closest to \(L(c)\). So we can make the following observation: If \(L(c)\) is not badly cut, then the detour through the portals when assigning \(c\) to \(f\) is fine (namely at most \(\varepsilon\) times the distance from \(L(c)\) to its closest facility in OPT). Otherwise, if \(L(c)\) is also badly cut, then we simply argue that there exists a near-optimal solution which contains \(L(c)\), in which case \(c\) is now served optimally at a cost of 0 (in the new instance).

**From bicriteria to opening exactly \(k\) centers:** Since \(L(c)\) is badly cut with probability \(O(\varepsilon^3)\), this leads to a solution opening \((1 + O(\varepsilon^3))k\) centers. At first, it looks difficult to then reduce the number of centers to \(k\) without increasing the cost of the solution by a factor larger than \((1 + \varepsilon)\). However, and perhaps surprisingly, we show in Lemma IV.6 that this can be avoided: we show that there exists a near-optimal solution that contains the badly cut centers of \(L(c)\).

We can then conclude that a near-optimal solution can be computed by a simple dynamic-programming procedure on the split-tree decomposition to identify the best solution in the modified instance.

Our result on Facility Location in Section III provides a simple illustration of these ideas — avoiding the bicriteria issue due to the hard bound on the number of opened facilities for the \(k\)-Median and \(k\)-Means problems. Our main result on \(k\)-Median and \(k\)-Means is described in Section IV. We discuss some extensions of the framework in Section V.

C. Related work

**On clustering problems:** The clustering problems considered in this paper are known to be NP-hard, even restricted to inputs lying in the Euclidean plane (see Mahajan et al. [29] or Dasgupta and Freund [16] for \(k\)-Means, Megiddo and Supowit [34] for the problems with outliers, and Masuyama et al. [31] for \(k\)-Center). The problems of Facility Location and \(k\)-Median have been studied since a long time in graphs, see e.g. [24]. The current best
approximation ratio for metric Facility Location is 1.488, due to Li [27], whereas it is 2.67 for k-Median, due to Byrka et al. [8].

The problem of k-Means in general graphs also received a lot of attention (see e.g., Kanungo et al. [24]) and the best approximation ratio is 6.357, due to Ahmadian et al. [1].

Clustering problems with outliers where first studied by Charikar et al. [10], who devised an (O(1), (1 + O(ε))-approximation for k-Median with outliers and a constant factor approximation for prize-collecting k-Median. More recently, Friggstad et al. [18] showed that local search provides a bicriteria approximation, where the number of centers is approximate instead of the number of outliers. However, the runtime is $n^{1/(ε,d)}$, and thus we provide a much faster algorithm. To the best of our knowledge, we present the first approximation scheme that preserves the number of centers.

The k-Center problem is known to be NP-hard to approximate within any factor better than 2, a bound that can be achieved by a greedy algorithm [17]. This is related to the problem of covering points with a minimum number of disks (see e.g. [28], [30]). Marx and Pilipczuk [30] proposed an exact algorithm running in time $n^{\sqrt{n}}+O(1)$ to find the maximum number of points covered by $k$ disks and showed a matching lower bound, whereas Liao et al. [28] presented an algorithm running in time $O(mn^{O(1/ε^2 \log^2 1/ε)})$ to find a $(1+\varepsilon)$-approximation to the minimal number of disks necessary to cover all the points (where $m$ is the total number of disks and $n$ the number of points). This problem is closely related to k-Center: the optimal value of k-Center on a set $V$ is the minimal number $L$ such that there exist $k$ disks of radius $L$ centered on points of $V$ covering all points of $V$. Hence, the algorithm from [28] can be directly extended to find a solution to k-Center with $(1+\varepsilon)k$ centers and optimal cost. Loosening on the approximation allows us to present a much faster algorithm.

On doubling dimension: Despite their hardness in general metrics, these problems admit a PTAS when the input is restricted to a low dimensional metric space: Friggstad et al. [19] showed that local search gives a $(1+\varepsilon)$-approximation. However, the running time of their algorithm is $n^{(d/ε)^{O(1/ε)}}$ in metrics with doubling dimension $d$.

A long line of research exists on filling the gap between results for Euclidean spaces and metrics with bounded doubling dimension. This started with the work of Talwar [37], who gave QPTASs for a long list of problems. The complexity for some of these problems was improved later on: for the Traveling Salesperson problem, Gottlieb [20] gave a near-linear time approximation scheme, Chan et al. [9] gave a PTAS for Steiner Forest, and Gottlieb [20] described an efficient spanner construction.

II. PRELIMINARIES

A. Definitions

Consider a metric space $(V, \text{dist})$. For a vertex $v \in V$ and an integer $r \geq 0$, we let $\beta(v, r) = \{ w \in V | \text{dist}(v, w) \leq r \}$ be the ball around $v$ with radius $r$. The doubling dimension of a metric is the smallest integer $d$ such that any ball of radius $2r$ can be covered by $2^d$ balls of radius $r$. We call $D$ the aspect-ratio (sometimes referred to as spread in the literature) of the metric, i.e., the ratio between the largest and the smallest distance.

Given a set of points called clients and a set of points called candidate centers in a metric space, the goal of the k-Median problem is to output a set of $k$ centers (or facilities) chosen among the candidate centers that minimizes the sum of the distances from each client to its closest center. More formally, an instance to the k-Median problem is a 4-tuple $(C, F, \text{dist}, k)$, where $(C \cup F, \text{dist})$ is a metric space and $k$ is a positive integer. The goal is to find a set $S \subseteq F$ such that $|S| \leq k$ and $\sum_{c \in C} \min_{f \in S} \text{dist}(c, f)$ is minimized. Let $n = |C \cup F|$. The k-Means problem is identical except from the objective function which is $\sum_{c \in C} \min_{f \in S} \text{dist}(c, f)^2$.

In the Facility Location problem, the number of centers in the solution is not limited but there is a cost $w_f$ for each candidate center $f$ and the goal is to find a solution $S$ minimizing $\sum_{c \in C} \min_{f \in S} \text{dist}(c, f) + \sum_{f \in S} w_f$.

For those clustering problems, it is convenient to name the center serving a client. For a client $c$ and a solution $S$, we denote $S(c)$ the center closest to $c$, and $S_c := \text{dist}(c, S(c))$ the distance to it.

In this paper, we consider the case where the set of candidate centers is part of the input. A variant of the k-Median and k-Means problems in Euclidean metrics allows to place centers anywhere in the space and specifies the input size as simply the number of clients. We note that up to losing a polylogarithmic factor in the running time, it is possible to reduce this variant to our setting by computing a set of candidate centers that approximate the best set of centers in $\mathbb{R}^d$ [32].

A $\delta$-net of $V$ is a set of points $X \subseteq V$ such that for all $v \in V$ there is an $x \in X$ such that $\text{dist}(v, x) \leq \delta$, and for all $x, y \in X$ we have $\text{dist}(x, y) \geq \delta$. A net is therefore a set of points not too close to each other, such that every point of the metric is close to a net point. The following lemma bounds the cardinality of a net in doubling metrics.

Lemma II.1 (from Gupta et. al [21]). Let $(V, d)$ by a metric space with doubling dimension $d$ and diameter $\Delta$, and let $X$ be a $\delta$-net of $V$. Then $|X| \leq 2^d \log (\Delta/\delta)$.}

Another property of doubling metrics that will be useful for our purpose is the existence of low-stretch spanners with a linear number of edges. More precisely, Har-El and Mendel [22] showed that one can find a graph (called a spanner) in the input metric that has $O(n)$ edges such that
distances in the graph approximate the original distances up to a constant factor. This construction takes time $2^{O(d)}n$.

We will make use of these spanners only for computing constant-factor approximations of our problems: for this purpose, we will therefore assume that the number of edges is $m = 2^{O(d)}n$.

We will also make use the following lemma.

**Lemma II.2 ([14]).** Let $p \geq 0$ and $1/2 < \epsilon > 0$. For any $a, b, c \in A \cup F$, we have $\text{dist}(a, b)^p \leq (1 + \epsilon)^p \text{dist}(a, c)^p + \text{dist}(c, b)^p(1 + 1/\epsilon)^p$.

### B. Decomposition of Metric Spaces

As pointed out in our techniques section, we will make use of hierarchical decompositions of the input metric. We define a hierarchical decomposition (sometimes simply a decomposition) of a metric $(V, \text{dist})$ as a collection of partitions $D = \{B_0, \ldots, B_{|D|}\}$ that satisfies the following:

- Each $B_i$ is a partition of $V$.
- $B_i$ is a refinement of $B_{i+1}$, namely for each part $B \in B_i$ there exists a part $B' \in B_{i+1}$ that contains $B$.
- $B_0$ contains a singleton set for each $v \in V$, while $B_{|D|}$ is a trivial partition that contains only one set, namely $V$.

We define the $i$th level of the decomposition to be the partition $B_i$, and call $B_i$ a level-$i$ part. If $B' \in B_{i-1}$ is such that $B' \subset B$, we say that $B'$ is a subpart of $B$.

For a given decomposition $D = \{B_0, \ldots, B_{|D|}\}$, we say that a vertex $u$ is cut from $v$ at level $j$ if $j$ is the maximum integer such that $v$ is in some $B \in B_j$ and $u$ is in some $B' \in B_j$ with $B \neq B'$. For a vertex $v \in F$ we say that the ball $\beta(v, 2^j)$ is cut by $D$ at level $j$ if there is at least one vertex of the ball that is cut from $v$ at level $j$.

A key ingredient for our result is the following lemma, that introduces some properties of the hierarchical decomposition (sometimes referred to as split-tree) proposed by Talwar [37] for low-doubling metrics.

**Lemma II.3 (Reformulation of [37], [6]).** For any metric $(V, \text{dist})$ of doubling dimension $d$ and any $\rho > 0$, there is a randomized hierarchical decomposition $D$ such that the diameter of a part $B \in B_i$ is at most $2^{i+1}$, $|D| \leq \lceil \log_2(\text{diam}(V)) \rceil$, and:

1. **Scaling probability:** for any $v \in V$, radius $r$, and level $i$, we have
   $$\Pr[D \text{ cuts } \beta(v, r) \text{ at a level } i] \leq 2^{d+2r/2}.$$

2. **Concise and precise portal set:** For any set $B \in B_i$ where $B_i \in D$, there is a set of portals $P_B$ such that,
   a) **concise:** $|P_B| \leq 1/\rho^2$; and
   b) **precise:** for any ball $\beta(v, r) \subseteq B$ cut by $C_T$ at level $i$ and pair of distinct sets $B_1, B_2 \in B_{i-1}$ on level $i - 1$, we have for any $u \in B_1 \cap \beta(v, r)$, and $w \in B_2 \cap \beta(v, r)$,
   $$\min_{p \in P_B} \{\text{dist}(u, p) + \text{dist}(p, w)\} \leq \text{dist}(u, w) + O(2^i).$$

Moreover, this decomposition can be found in time $(1/\rho)^{O(d)}n \log \Delta$.

### C. Formal Definition of Badly Cut Vertices

As sketched in the introduction, the notion of badly cut lies at the heart of our analysis. We define it formally here. We denote $\kappa(\epsilon, p) = \frac{\epsilon^d}{p \tau(p, \epsilon)}$ and $\tau(\epsilon, d) = 2d + 2 + \log \log(1/\epsilon) + \log(1/\kappa(\epsilon, p))$, two parameters that are often used throughout this paper.

**Definition II.4.** Let $(V, \text{dist})$ be a doubling metric, let $D$ be a hierarchical decomposition on $(V, \text{dist})$, and $\epsilon > 0$. A client $v$ is badly cut w.r.t. $D$ if there exists an integer $i$ such that $2^i \in [\epsilon L_v, L_v/\epsilon]$ and $\beta(2^i, 2^j)$ is cut at some level $j$ greater than $i + \tau(\epsilon, d)$.

Similarly, a center $f$ of $L$ is badly cut w.r.t $D$ if there exists an integer $i$ such that $2^i \in [\epsilon \text{OPT}_f, \text{OPT}_f/\epsilon]$ and $\beta(2^i, 2^j)$ is cut at some level $j$ greater than $i + \tau(\epsilon, d)$, where $\text{OPT}_f$ is the distance from $f$ to the closest facility of $\text{OPT}$.

In the following, when $D$ is clear from the context we simply say badly cut. The following lemma bounds the probability of being badly cut.

**Lemma II.5.** Let $(C \cup F, \text{dist})$ be a metric, and $D$ a random hierarchical decomposition given by Lemma II.3. Let $v$ be a vertex in $C \cup F$. The probability that $v$ is badly cut is at most $\kappa(\epsilon, p)$.

**Proof:** Consider first a vertex $v \in C$. By Property 1, the probability that a ball $\beta(v, 2^i)$ is cut at level $j$ is at most $2^{2d+2j}/2^j$. Hence the probability that a ball $\beta(v, 2^i)$, where $2^i \in [\epsilon^2 L_v, L_v/\epsilon^2]$, is cut at a level $j$ greater than $i + \delta + \log \log(1/\epsilon) + \log(1/\kappa(\epsilon, p))$ is at most $\frac{\epsilon^d}{p \tau(p, \epsilon)}$. Taking a union bound over all balls of radius $2^i$ such that $i$ is an integer and $2^i \in [\epsilon^2 L_v, L_v/\epsilon^2]$ we have that the probability that $v$ is badly cut is at most $4 \log(1/\epsilon) \frac{\kappa(\epsilon, p)}{4 \log(1/\epsilon)} = \kappa(\epsilon, p)$.

The proof for $v \in F$ is identical.

### D. Preprocessing

In the following, we will work with the slightly more general version of the clustering problems where there is some demand on each vertex: there is a function $\chi : C \mapsto \{1, \ldots, n\}$ and the goal is to minimize $\sum_{c \in C} \chi(c) \cdot \min_{f \in F} \text{dist}(c, f) + \sum_{f \in F} w_f$ for the Facility Location problem, or $\sum_{c \in C} \chi(c) \cdot \min_{f \in F} \text{dist}(c, f)$ and $\sum_{c \in C} \chi(c) \cdot \min_{f \in F} \text{dist}(c, f)^2$ for $k$-Median and $k$-Means respectively. This also extends to any $\sum_{c \in C} \chi(c) \cdot \min_{f \in F} \text{dist}(c, f)^p$ with constant $p$. 

545
We will preprocess the input instance to transform it into several instances of the more general clustering problem, ensuring that the aspect-ratio $\Delta$ of each instance is polynomial. We defer this construction to Appendix A.

III. A Near-Linear Time Approximation Scheme for Non-Uniform Facility Location

To demonstrate the utility of the notion of badly cut, we show how to use it to get a near-linear time approximation scheme for Facility Location in metrics of bounded doubling dimension. In this context we refer to centers in the set $F$ of the input as facilities.

We first show a structural lemma that allows to focus on instances that do not contain any badly cut client. Then, we prove that these instance have portal-respecting solutions that are nearly optimal, and that can be computed with a dynamic program. We conclude by providing a fast dynamic program, that takes advantage of all the structure provided before.

A. Structural Lemma

Let $\epsilon > 0$, and consider a metric space $(V, \text{dist})$ and an instance $I$ of the Facility Location problem on $(V, \text{dist})$. Namely, an instance whose client and candidate center sets are subsets of $V$. Our first step is to show that, given $I$, a randomized decomposition $D$ of $(V, \text{dist})$ and any solution $L$ for $I$ on $(V, \text{dist})$, we can build an instance $I_D$ such that any solution $S$ has a similar cost in $I$ and in $I_D$, and more importantly $I_D$ does not contain any badly cut client with respect to $D$. The definition of $I_D$ depends on the randomness of $D$. Define $B_D$ be the set of badly cut facilities of $L$ w.r.t $D$.

Let $\text{cost}_{I_D} : V \rightarrow \mathbb{R}$ be a function that given a set of centers in an instance $I_D$ on $(V, \text{dist})$, returns the $k$-Median cost induced by the set of centers in $I_D$. For any instance $I_D$ on $(V, \text{dist})$, we let

$$\nu_{I_D} = \max_{\text{solution } S} \left( \text{cost}_{I_D}(S) - (1 + 3\epsilon) \text{cost}_{I_D}(S) \right).$$

We say that an instance $I_D$ has small distortion w.r.t. $I$ if $\sum_{f \in B_D} w_f \leq \epsilon \text{cost}(L)$ and $\nu_{I_D} \leq \epsilon \text{cost}(L)$. When $I$ is clear from the context we simply say that $I_D$ has small distortion.

In the following, we will always work with a particular $I_D$ constructed from $I$ and a precomputed approximate solution $L$ as follows: $I$ is transformed such that every badly cut client $c$ is moved to $L(c)$, namely, there is no more client at $c$ in $I_D$ but an extra client is added at $L(c)$. All the other clients stay as they are.

What we would like to prove is that the optimal solution in $I$ can be transformed to a solution in $I_D$ with a small additional cost, and vice versa. The intuition behind this is the following: a client of the solution $L$ is badly cut with probability $\kappa(\epsilon, p)$ (from Lemma II.5), hence every client contributes with $\kappa(\epsilon, p)L_c$, to transform any solution $S$ for the instance $I$ to a solution for the instance $I_D$, and vice versa.

However, we will need to convert a particular solution in $I_D$ (think of it as $\text{OPT}_{I_D}$) to a solution in $I$: this particular solution depends in the randomness of $D$, and this short argument does not apply because of dependency issues. It is nevertheless possible to prove that $I_D$ has a small distortion, as done in the following lemma.

Lemma III.1. Given an instance $I$ of Facility Location, a randomized decomposition $D$ and a solution $L$, let $I_D$ be the instance obtained from $I$ by moving every badly cut client $c$ to $L(c)$ (as described above). The probability that $I_D$ has small distortion is at least $1 - \epsilon$.

Proof: To show the lemma, we will show that $\mathbb{E}[\nu_{I_D}] \leq \epsilon^2 \text{cost}(L)/2$. Then, by Markov’s inequality and taking a union bound over the probabilities of failure yields the lemma. Note that $\mathbb{E}[\sum_{f \in B_D} w_f] = \sum_{f \in L} \text{Pr}[f \text{ badly cut}] \cdot w_f \leq \epsilon^2 \text{cost}(L)/2$ is immediate from Lemma II.5.

We thus aim at showing that $\mathbb{E}[\nu_{I_D}] \leq \epsilon^2 \text{cost}(L)/2$. In the sake of lightening equations, we will note $\sum_{c \in bcc.}$ the sum over all badly cut clients $c$.

By definition, we have that for any solution $S$,

$$\text{cost}(S) - \text{cost}_{I_D}(S) \leq \sum_{c \in bcc.} \text{dist}(c, S)^p - \text{dist}(S, L(c))^p$$

$$\leq \sum_{c \in bcc.} \left( (1 + 3\epsilon) \text{dist}(S, L(c))^p \right.$$

$$+ \frac{\text{dist}(c, L(c))^p}{(\epsilon/p+1)^p} - \text{dist}(S, L(c))^p) \right).$$

using Lemma II.2 with parameter $\epsilon/p$. This is equal to

$$\sum_{c \in bcc.} 3\epsilon \cdot \text{dist}(S, L(c))^p + \frac{\text{dist}(c, L(c))^p}{(\epsilon/p+1)^p},$$

and so we have

$$\text{cost}(S) - (1 + 3\epsilon) \text{cost}_{I_D}(S) \leq \sum_{c \in bcc.} \frac{\text{dist}(c, L(c))^p}{\epsilon/p+1}.$$
and we conclude
\[(1 - 3\varepsilon)\text{cost}_{\mathcal{I}}(S) - \text{cost}(S) \leq \sum_{b,e, c} \frac{\text{dist}(e, L(c))^p}{(\varepsilon/(p + 1))^p} \]

Therefore, the expected value of \(\nu_{\mathcal{I}}\) is
\[E[\nu_{\mathcal{I}}] \leq \sum_{\text{client } e} P_e[c \text{ badly cut}] \cdot \frac{\text{dist}(e, L(c))^p}{(\varepsilon/(p + 1))^p}.\]

Applying Lemma II.5 and using \(\kappa(\varepsilon, p) = \frac{e^{p+2}}{(p+1)^p}\), we conclude \(E[\nu_{\mathcal{I}}] \leq \varepsilon^2 \cdot \text{cost}(L)\). The lemma follows for a sufficiently small \(\varepsilon\).

**B. Portal Respecting Solution**

In the following, we fix an instance \(\mathcal{I}\), a decomposition \(\mathcal{D}\) and a solution \(L\). By Lemma III.1, \(\mathcal{I}\) has small distortion with probability at least \(1 - \varepsilon\) and so we condition on this event from now on.

We explore the structure that this conditioning gives to the solution. We will show that there exists a solution \(\text{OPT}'\) with small cost such that each client \(c\) is cut from its serving facility \(f\) at a level at most \(\log(5(L_c + OPT_c)) + \varepsilon(d, \varepsilon)\). This allows one to consider portal-respecting solution, where every client to facility path goes in and out parts of the decomposition only at designated portals. Indeed, the detour incurred by making a portal path respecting depends on the level where its extremities are cut: more precisely, it is an epsilon fraction of the diameter at that level. Hence, ensuring that this level stays small implies that the detour made is small (in our case, \(O(\varepsilon(L_c + OPT_c))\)). Such a solution can be computed by a dynamic program that we will present afterwards.

In the following, we consider the solution \(\text{OPT}' = \text{OPT} \cup B\) (where OPT is the optimal solution for the instance \(\mathcal{I}\)). Recall that \(L_c\) and \(\text{OPT}\) are the distances from the original position of \(c\) to \(L\) and \(\text{OPT}\), but \(c\) may have been moved to \(L(c)\) and \(B\) is the set of badly cut facilities of \(L\) w.r.t. \(\mathcal{D}\).

**Lemma III.2.** Let \(\mathcal{I}\) be an instance of Facility Location with a randomized decomposition \(\mathcal{D}\), and \(L\) be a solution for \(\mathcal{I}\), such that \(\mathcal{I}\) has small distortion. For any client \(c\) in \(\mathcal{I}\), let \(\text{OPT}'(c)\) be the closest facility to \(c\) in \(\text{OPT}\). Then \(c\) and \(\text{OPT}'(c)\) are separated at a level at most \(\log(5(L(c) + OPT(c))) + \varepsilon(d, \varepsilon)\).

**Proof:** Let \(c\) be a client. To find the level at which \(c\) and \(\text{OPT}'(c)\) are separated, we distinguish between two cases: either \(c\) in \(\mathcal{I}\) is badly cut w.r.t. \(\mathcal{D}\), or not.

If \(c\) is badly cut, then it is now located at \(L(c)\) in the instance \(\mathcal{I}\). In this case, either:

1) \(L(c)\) is also badly cut, and therefore \(L(c) \in B\) \(\subseteq \text{OPT}'\) and so \(\text{OPT}'(c) = L(c)\). It follows that \(c\) and \(\text{OPT}'(c)\) are never separated.

2) \(L(c)\) is not badly cut. Then \(\text{dist}(c, \text{OPT}'(c)) \leq \text{OPT}_L(c)\). We bound the level at which \(c\) and \(\text{OPT}'(c)\) are separated. Since \(L(c)\) is not badly cut, \(\text{Definition II.4}\) implies that \(L(c)\) and \(\text{OPT}(L(c))\) are cut at a level at most \(\log(\text{OPT}_L(c)) + \varepsilon(d, \varepsilon)\). By triangle inequality, \(\text{OPT}_L(c) = \text{dist}(L(c), \text{OPT}(L(c))) \leq L_c + \text{OPT}_c\), and thus \(c\) and \(\text{OPT}'(c)\) are also separated at a level at most \(\log(L_c + \text{OPT}_c) + \varepsilon(d, \varepsilon)\).

We now turn to the case where \(c\) is not badly cut. In which case, \(c\) is not moved to \(L_c\) and the balls \(\beta(c, 2^i)\) with \(2^i \in [\varepsilon L_c, L_c/\varepsilon]\) are not badly cut. We make a case distinction according to \(\text{OPT}_c\).

1) If \(L_c \leq \varepsilon \text{OPT}_c\), then we have the following. If \(L(c)\) is badly cut, \(L(c)\) is open and therefore \(\text{OPT}'_c = L_c\). Moreover, since \(c\) is not badly cut the ball \(\beta(c, L_c)\) is cut at level at most \(\log(L_c) + \varepsilon(d, \varepsilon)\). Therefore \(c\) and \(\text{OPT}'(c)\) are separated at level at most \(\log(L_c) + \varepsilon(d, \varepsilon)\).

In the case where \(L(c)\) is not badly cut, both \(c\) and \(\text{OPT}'(c)\) lie in the ring centered at \(L(c)\) and of diameter \(\text{2OPT}_L(c)\).

Indeed,
\[\text{dist}(c, L(c)) \leq \text{dist}(c, \text{OPT}'(c)) \leq \text{dist}(c, \text{OPT}(L(c))) \leq \varepsilon(d, c, L(c)) + \text{dist}(L(c), \text{OPT}(L(c)))\]

And therefore, for any \(\varepsilon \leq 2/3\),
\[\text{dist}(c, L(c)) \leq \varepsilon(d, \text{OPT}'(c)) \leq \varepsilon(d, \text{OPT}(L(c))) \leq \text{dist}(c, \text{OPT}(L(c))) + \text{dist}(L(c), \text{OPT}(L(c))) \leq \left(1 + \frac{2\varepsilon}{1 - \varepsilon}\right) \text{OPT}_L(c),\]

which is smaller than \(\text{2OPT}_L(c)\) for any \(\varepsilon \leq 1/3\). Hence we have \(c, \text{OPT}'(c) \in \beta(L(c), 2\text{OPT}_L(c))\). To apply the definition of badly cut, we need to consider rings with radius power of \(2\); let us therefore consider \(i\) such that \(2^i \leq 4\text{OPT}_L(c)\).

Since \(L(c)\) is not badly cut, this ring is not cut by a too high level part either. Therefore \(c\) and \(\text{OPT}'(c)\) are separated at level at most \(i + \varepsilon(d, \varepsilon)\), which is at most \(\log(4\text{OPT}_L(c)) + \varepsilon(d, \varepsilon)\).

Now, since \(\text{OPT}_L(c) \leq \text{dist}(L(c), \text{OPT}(c)) \leq \text{dist}(L(c), \text{OPT}(c)) + \text{dist}(\text{OPT}(c), \text{OPT}(c)) \leq (1 + \varepsilon)\text{OPT}_c\), we have that \(\log(4\text{OPT}_L(c)) \leq \log(5\text{OPT}_c)\), and hence \(c\) and \(\text{OPT}'(c)\) are separated at level at most \(\log(5\text{OPT}_c) + \varepsilon(d, \varepsilon)\).

2) If \(L_c \geq \varepsilon \text{OPT}_c/\varepsilon\) then, since \(c\) is not badly cut, the ball centered at \(c\) and of radius \(\varepsilon L_c\) is not badly cut. Since we have \(\text{dist}(c, \text{OPT}'(c)) \leq \varepsilon(d, \text{OPT}_c) \leq \varepsilon L_c\), \(c\) and \(\text{OPT}'(c)\) lie in the ball \(\beta(c, \varepsilon L_c)\) and are thus cut at level at most \(\log(\varepsilon L_c) + \varepsilon(d, \varepsilon)\).

3) If \(\varepsilon L_c \leq \text{OPT}_c \leq L_c/\varepsilon\), then since \(c\) is not badly cut the ball \(\beta(c, \text{OPT}(c))\) is cut at level at most \(\log(2\text{OPT}_c) + \varepsilon(d, \varepsilon)\). Moreover, \(\text{OPT}'(c)\) lies in this ball.
This concludes the proof.

A path between two nodes \( u \) and \( v \) is a sequence of nodes \( w_1, \ldots, w_k \) with \( u = w_1, v = w_k \), and its length is \( \sum \text{dist}(w_j, w_{j+1}) \). A solution to the problem can therefore be seen as a set of facility, together with a path for each client that connects it to a facility. We say a path is portal-respecting if it enters and leaves parts of the decomposition \( D \) only at portals. More precisely, for every pair \( w_j, w_{j+1} \) of the sequence, if \( w_j \) and \( w_{j+1} \) lie in different parts of some level \( i \), then these nodes are also portals at this level (note that such a path is guaranteed to exist, since we assume that portals are nested; cf. Lemma II.3). We define a portal-respecting solution to be a solutions such that each path from a client to its closest facility in the solution is portal-respecting.

The dynamic program will compute an optimal portal-respecting solution. Therefore, we need to prove that the optimal portal-respecting solution is close to the optimal solution. Let \( u \) and \( v \) be two vertices separated at level \( i \) by the decomposition \( D \). We note a property of the decomposition that will simplify our calculations. For the path between \( u \) and \( v \) to be portal-respecting, there needs to be a detour at every level below \( i \), with an error of at most \( \sum_{j \leq i} O(\rho^{2j}) \leq O(\rho^{2i+1}) \). This error comes from the preciseness property in Lemma II.3. In the remainder of the paper, we will thus bound the total error incurred across all levels by \( O(\rho \cdot 2^i) \), where \( i \) is the level at which \( u \) and \( v \) are separated. We let \( \rho = \varepsilon^{\rho - \tau(c,d)} \), and for a solution \( S \) define \( B(S) := \sum_{c, i : c \in S|v| \text{ cut at level } i} \varepsilon^{2i}. \) One can see \( B(S) \) as a budget, given by the fact that vertices are not badly cut.

**Lemma III.3.** Given an instance \( I \) and a solution \( L \), it holds with probability \( 1 - \varepsilon \) (over \( D \)) that there exists a portal-respecting solution \( S \) in \( I \) such that \( \text{cost}(S) \leq \text{cost}(L) \).

**Proof:** From Lemma I.1, with probability \( 1 - \varepsilon \) it holds that the instance \( I_D \) has small distortion. We condition now on this event. Consider solution \( \text{OPT}' \). Since \( I_D \) has small distortion, we have that the facility cost of \( \text{OPT}' \) is at most the facility cost of \( \text{OPT} \) plus \( \varepsilon \text{cost}(L) \). Furthermore, again since \( I_D \) has small distortion we have that \( \text{cost}(\text{OPT}) \leq \text{cost}(\text{OPT}) \).

We then bound the cost of making \( \text{OPT}' \) portal respecting by applying Lemma III.2. Since each client \( c \) of \( I_D \) is separated from \( \text{OPT}'(c) \) at level at most \( \log(5(L_c + \text{OPT}_c)) + \tau(c,d) \), we have that the detour for making the assignment of \( c \) to \( \text{OPT}'(c) \) portal-respecting is at most \( \rho^{\tau(c,d)}(5(L_c + \text{OPT}_c)) \). Choosing \( \rho = \varepsilon^{\rho - \tau(c,d)} \) ensures that the detour is at most \( O(\varepsilon(L_c + \text{OPT}_c)) \). This also bounds \( B(\text{OPT}') \leq O(\varepsilon)(\text{cost}(L) + \text{cost}(\text{OPT})) \).

Therefore, taking \( S = \text{OPT}' \) ensures that
\[
\text{cost}(S) \leq \text{cost}(\text{OPT}) + 20c(\varepsilon \text{cost}(\text{OPT}) + \varepsilon \text{cost}(L)) \\
\leq (1 + O(\varepsilon)) \text{cost}(\text{OPT}) + O(\varepsilon \text{cost}(L))
\]

**C. The Algorithm**

Using Lemmas A.1 and A.2, we can assume that the aspect-ratio of the instance is \( O(\varepsilon^2/\varepsilon) \). Our algorithm starts by computing a constant-factor approximation \( L \), using Meyerson’s algorithm [35]. It then computes a hierarchical decomposition \( D \), as explained in the Section II-B, with parameter \( \rho = \varepsilon^{2 - \tau(c,d)} \).

Given \( L \) and the decomposition \( D \), our algorithm finds all the badly cut clients as follows. For each client \( c \), determine whether \( c \) is badly cut or not, only \( O(\log(1/\varepsilon)) \) balls have to be considered, namely the balls centered at \( c \) and with exponentially growing radius in \( [c, L_c, L_c/\varepsilon] \). For each such ball \( \beta \), the algorithm checks whether the decomposition cuts \( \beta \) at a level that is too high, making \( c \) badly cut. This can be done efficiently by verifying whether \( c \) is at distance smaller than \( L_c/\varepsilon \) to such a part of too high level. Thus, the algorithm finds all the badly cut clients in near-linear time.

The next step of the algorithm is to compute instance \( I_D \) by moving every badly cut client \( c \) to its facility in \( L \). This can also be done in linear time.

A first attempt at a dynamic program.: We now turn to the description of the dynamic program (DP) for obtaining the best portal-respecting solution of \( I_D \). This is the standard dynamic program for Facility Location and we only describe it for the sake of completeness, the reader familiar with this can skip to the analysis.

There is a table entry for each part of the decomposition, and two vectors of length \( n_p \), where \( n_p \) is the number of portals in the part (we call such a triplet a configuration). Each configuration given by a part \( R \) and vectors \( (\ell_1, \ldots, \ell_p) \) and \( (s_1, \ldots, s_p) \) (called the portal parameters), encodes a possible interface between part \( R \) and a solution for which the \( r \)th portal has approximate distance \( \ell_r \) to the closest facility inside of \( R \), and approximate distance \( s_r \) to its closest facility outside of \( R \). The value stored for such a configuration in a table entry is the minimal cost for a solution with facilities respecting the constraints induced by the vectors on the distances between the solution and the portals inside the part (as described below).

To fill the table, we use a dynamic program following the lines of Arora et al. [4] or Kolliopoulos and Rao [25]. If a part has no descendant (meaning the part contains a single point), computing the solution given the configuration is straightforward: either a center is opened on this point or not, and it is easy to check the consistency with the configuration, where only the distances to portals inside the part need to be verified. At a higher level of the decomposition, a
solution is simply obtained by going over all the sets of parameter values for all the children parts. It is immediate to see whether sets of parameter values for the children can lead to a consistent solution:

- for each portal $p_1$ of the parent part, there must be one portal $p_2$ of a child part such that the distance from $p_1$ to a center inside the part prescribed by the configuration corresponds to $\text{dist}(p_1, p_2)$ plus the distance from $p_2$ to a center inside the child part;
- for each portal $p_2$ of a child part, there must exist either:
  - a portal $p_1$ of the parent part such that the distance from $p_2$ to a center outside its part prescribed by the configuration is $\text{dist}(p_1, p_2)$ plus the distance from $p_1$ to a center outside of the part,
  - or a portal $p_1$ of another child part such that this distance is $\text{dist}(p_1, p_2)$ plus the distance from $p_1$ to a center inside the child part.

Recall that the aspect ratio is $n^{O(1)}$, and so this dynamic program has a complexity polylogarithmic in $n$, since there are $O(\log n)$ possible values for a rounded distance. However, using the budget given by Lemma III.3, one can shave off the logarithmic factors.

A faster dynamic program: We now describe a faster dynamic program. Consider a level where the diameter of the parts is say $\Delta$. Each configuration is again given by a part $R$ and portal parameters $(\ell_1, \ldots, \ell_n)$ and $(s_1, \ldots, s_n)$, but with the restriction that $\ell_i$ and $s_i$ are multiples of $\varepsilon \Delta$ in the range $[0, \Delta/\varepsilon + \Delta]$. A flag is additionally attached to the configuration whose meaning will be explained shortly.

We sketch here the intuition behind this restriction. Since the diameter of the part is $\Delta$ we can afford a detour of $\varepsilon \Delta$, that is taken into account in the budget $B(S)$. Hence, distances can be rounded to the closest multiple of $\varepsilon \Delta$.

Now, suppose that the closest facility outside the part is at distance greater than $\Delta/\varepsilon$, and that there is no facility inside the part. Then, since the diameter is $\Delta$, up to losing an additive $\varepsilon \Delta$ in the cost of the solution computed, we may assume that all the points of the part are assigned to the same facility. So the algorithm is not required to have the precise distance to the closest center outside the part, and it uses the flag to reflect that it is in this regime. We can then treat this whole part as a single client (weighted by the number of clients inside the part) to be considered at higher levels. Assuming that the closest facility is at distance less than $\Delta/\varepsilon$, we have that for any portal of the part the closest facility is at distance at most $\Delta/\varepsilon + \Delta$ (since $\Delta$ is the diameter of the part).

On the other hand, if there is some facility inside the part and the closest facility outside the part is at distance at least $\Delta/\varepsilon$, then each client of the part should be served by a facility inside the part in any optimal assignment. Thus it is not necessary that the algorithm iterates over configurations where the distances outside the part are more than $\Delta/\varepsilon$; it is enough to do it once and use the answer for all other queries.

Analysis – Proof of Theorem I.3: The two following lemmas show that the solution computed by this algorithm is a near-optimal one, and that the complexity is near-linear: this proves Theorem I.3.

Lemma III.4. Let $S$ be as in Lemma III.3. The algorithm computes a solution $S^*$ with cost at most $\text{cost}_{\varepsilon \Delta}(S^*) \leq (1 + O(\varepsilon))\text{cost}_{\varepsilon \Delta}(S) + B(S)$. Proof: We show that the solution $S$ can be adapted to a configuration of the DP with and extra cost $B(S)$. For this, let $c$ be a client served by a facility $S(c)$, and let $p_1, \ldots, p_k$ be the portal-respecting path from $c$ to $S(c)$, $p_i$ being a portal at level $l_i$, with $p_1 = c$ and $p_k = S(c)$. The cost of $c$ in $S$ is therefore $\sum d(p_i, p_{i+1})$.

The distance between $c$ and $S(c)$ is approximated at several place of the DP:

- When $d(p_i, S(c)) \leq 2^i/\varepsilon + 2^i$, the distance between $p_i$ and $S(c)$ is rounded to the closest multiple of $\varepsilon 2^i$, incurring a cost difference of $\varepsilon 2^i$.
- When $d(p_i, S(c)) \geq 2^i/\varepsilon + 2^i$, the whole part is contracted and served by a single facility at distance at least $2^i/\varepsilon$. The cost difference for client $c$ is therefore $2^i \leq \varepsilon d(p_i, S(c))$. Since the diameters of the parts are geometrically increasing, the total cost difference for such operations is bounded by $2\varepsilon d(p_i, S(c))$, where $l_i$ is the highest level where $d(p_j, S(c)) \geq 2^j/\varepsilon + 2^j$. This cost verifies $2\varepsilon d(p_i, S(c)) \leq 2\varepsilon \sum d(p_i, p_{i+1})$.

Hence, summing over all clients, the additional cost incurred by the DP compared is at most $B(S) + 2\varepsilon \text{cost}_{\varepsilon \Delta}(S)$. Since it computes a solution with minimal cost, it holds that $\text{cost}_{\varepsilon \Delta}(S^*) \leq (1 + 2\varepsilon)\text{cost}_{\varepsilon \Delta}(S) + B(S)$. ■

Corollary III.5. Let $S^*$ be the solution computed by the algorithm. With probability $1 - \varepsilon$, it holds that $\text{cost}_2(S^*) = (1 + O(\varepsilon))\text{cost}_2(\text{OPT}) + O(\varepsilon \text{cost}_2(L))$

Proof: Lemma III.3 ensures that, with probability $1 - \varepsilon$, the cost of $S$ is at most $(1 + O(\varepsilon))\text{cost}_2(\text{OPT}) + O(\varepsilon \text{cost}_2(L))$. Since $L$ is a constant factor approximation, this cost turns out to be $(1 + O(\varepsilon))\text{cost}_2(\text{OPT})$. Combining this with Lemma III.4 concludes the proof:

$$\text{cost}_2(S^*) = (1 + O(\varepsilon))\text{cost}_{\varepsilon \Delta}(S^*) = (1 + O(\varepsilon))(\text{cost}_{\varepsilon \Delta}(S) + B(S)) \leq (1 + O(\varepsilon))\text{cost}_2(\text{OPT}) + O(\varepsilon \text{cost}_2(L))$$ ■
Lemma III.6. This algorithm has complexity 
\[ \left( \frac{1}{\varepsilon} \right)^{2O(d \log d) / \varepsilon} \cdot n + 2^O(d) n \log n. \]

Proof: The preprocessing step (computing L, the hierarchical decomposition \( D \), and the instance \( I_D \)) has a running time \( O(n \log n) \), as all the steps can be done with this complexity: a fast implementation of Meyerson’s algorithm [35] tailored to graphs can compute \( L \) in time \( O(m \log n) \). Using it on the spanner computed with [22] gives a \((1)\)-approximation in time \( O(n \log n) \). As explained earlier, the hierarchical decomposition \( D \) and the instance \( I_D \) can also be computed with this complexity.

The DP has a linear time complexity: in a part of diameter \( \Delta \), the portal set is a \((12 / \varepsilon, d)\)-net, and hence has size \( 2^{\log(2^{\log(2^{d^2})}) / \varepsilon} \) by Lemma II.1. Since \( \tau(\varepsilon, d) = O(\log d) + 2 \log \left(\frac{\log(2 \cdot d)}{\varepsilon}\right)\), this number can be simplified to \( 2^{\log(2^{d^2} / \varepsilon)} \). Since each portal stores a table that can take only one \( 1 / \varepsilon^2 \) values, there is at most \( E = (1 / \varepsilon^2)^{2^{\log(2^{d^2})} / \varepsilon} = (1 / \varepsilon)^{2^{\log(2^{d^2})} / \varepsilon} \) possible table entries for a given part.

To fill the table, notice that a part has at most \( 2^O(d) \) children, due to the properties of the hierarchical decomposition. Going over all the sets of parameter values for all the children parts therefore takes time \( E^{2^O(d)} = (1 / \varepsilon)^{2^{\log(2^{d^2})} / \varepsilon} \). This dominates the complexity of the dynamic program, which is therefore \( n(1 / \varepsilon)^{2^{\log(2^{d^2})} / \varepsilon} \).

The total complexity of the algorithm is thus
\[ \left( \frac{1}{\varepsilon} \right)^{2^{\log(2^{d^2})} / \varepsilon} \cdot n + 2^O(d) n \log n \]

A. Towards a Structured Near-Optimal Solution

Let OPT be an optimal solution to \( \mathcal{I} \). We consider the mapping of the facilities of OPT to \( L \) defined as follows: for any \( f \in \text{OPT} \), let \( L(f) \) denote the facility of \( L \) that is the closest to \( f \). Recall that for a client \( c \), \( L(c) \) is the facility serving \( c \) in \( L \).

For any facility \( \ell \) of \( L \), define \( \psi(\ell) \) to be the set of facilities of OPT that are mapped to \( \ell \), namely, \( \psi(\ell) = \{ f \in \text{OPT} \mid L(f) = \ell \} \). Define \( L^1 \) to be the set of facilities of \( L \) for which there exists a unique \( f \in \text{OPT} \) such that \( L(f) = \ell \), namely \( L^1 = \{ \ell \mid |\psi(\ell)| = 1 \} \). Let \( L^0 = \{ \ell \mid |\psi(\ell)| = 0 \} \), and \( L^{\geq 2} = L - (L^1 \cup L^0) \).

Similarly, define \( \text{OPT}^1 = \{ f \in \text{OPT} \mid |L(f)\in L^1\} \) and \( \text{OPT}^{\geq 2} = \{ f \in \text{OPT} \mid L(f) \in L^{\geq 2} \} \). Note that \( |\text{OPT}^{\geq 2}| = |L^0| + |L^{\geq 2}| \), since \( |\text{OPT}^1| = |L^1| \) and, w.l.o.g., \( |\text{OPT}| = |L| = k \).

The construction of a structured near-optimal solution is made in 3 steps. The first one defines a solution \( \text{OPT}' \) as follows. Start with \( \text{OPT}' = \text{OPT} \).

- **Step 1.** Among the facilities of \( \text{OPT}^{\geq 2} \) that are not the closest of their corresponding facility in \( L^{\geq 2} \), remove from \( \text{OPT}' \) the subset \( \mathcal{H} \) of size \( \lfloor \varepsilon \cdot |\text{OPT}^{\geq 2}| / 2 \rfloor \) that yields the smallest cost increase.

This step makes room to add the badly cut facilities without violating the constraint on the maximum number of centers, while at the same time ensures that \( S^* \) has near-optimal cost, as the following lemma shows.

Lemma IV.1. After step 1, \( \text{OPT}' \) has cost \( (1 + O(\varepsilon)) \text{cost}(\text{OPT}) + O(\varepsilon) \text{cost}(L) \)

Proof: We show that the cost increase is at most \((1 + O(\varepsilon)) \text{cost}(\text{OPT}) + O(\varepsilon) \text{cost}(L)\).

Let \( \mathcal{H} \subseteq \text{OPT}^{\geq 2} \) be the set of facilities of \( \text{OPT}^{\geq 2} \) that are not the closest to their corresponding facility in \( L^{\geq 2} \), i.e., \( f \in \mathcal{H} \) if and only if \( f \notin \psi(\ell) \) for some \( \ell \in L^{\geq 2} \) and \( \text{dist}(f, \ell) > \min_{f' \in \psi(\ell)} \text{dist}(f', \ell) \) (breaking ties arbitrarily). The only elements in \( \text{OPT}^{\geq 2} - \mathcal{H} \) are the ones closest to their corresponding facilities. Hence for every facility of \( L^{\geq 2} \), \( |\psi(f)| \geq 2 \) there is therefore exactly one facility in \( \text{OPT}^{\geq 2} - \mathcal{H} \) and at least two in \( \text{OPT}^{\geq 2} \); and if \( |\psi(f)| = 0 \) then \( f \) does not correspond to any facility at all in \( \text{OPT}^{\geq 2} \). Therefore \( |\mathcal{H}| \geq |\text{OPT}^{\geq 2}| / 2 \).

We claim that for a client \( c \) served by \( f \in \mathcal{H} \) in the optimum solution OPT, i.e., \( f = \text{OPT}(c) \), the detour entailed by the deletion of \( f \) is \( O(\text{OPT}(c) + L(c)) \). Indeed, let \( f' \) be the facility of OPT that is closest to \( L(f) \), and recall that \( L(c) \) is the facility that serves \( c \) in the solution \( L \). Since \( f' \notin \mathcal{H} \), the cost to serve \( c \) after the removal of \( f \) is at most \( \text{dist}(c, f') \), which can be bounded by \( \text{dist}(c, f') \leq \text{dist}(c, f) + \text{dist}(f, L(f)) + \text{dist}(L(f), f') \). But by definition of \( f' \), \( \text{dist}(f', L(f)) \leq \text{dist}(L(f), f) \), and by definition of the function \( L \) we have \( \text{dist}(L(f), f) \leq \text{dist}(L(c), f) \),
so that $\text{dist}(c, f') \leq \text{dist}(c, f) + 2\text{dist}(f, L(c))$. Using the triangle inequality finally gives $\text{dist}(c, f') \leq \text{dist}(c, f) + 2\text{dist}(f, L(c))$ which is $O(\text{OPT}_f + L_c)$. For a facility $f$ of OPT, we denote $C(f)$ the set of clients served by $f$, i.e. $C(f) = \{c \in C \mid \text{OPT}(c) = f\}$. The total cost incurred by the removal of $f$ is then $O(\text{cost}_{\text{OPT}}(C(f)) + \text{cost}_L(C(f)))$.

Recall that in Step 1 we remove the set $\mathcal{H}$ of size $\lfloor \epsilon/\text{OPT}^2 \rfloor$ from $\mathcal{H}$, such that $\mathcal{H}$ minimizes the cost increase. We use an averaging argument to bound the cost increase: the sum among all facilities $f \in \mathcal{H}$ of the cost of removing the facility $f$ is less than $O(\text{cost}(\text{OPT}) + \text{cost}(L))$, and $|\mathcal{H}| = O(1/\epsilon) \cdot \lfloor \epsilon/\text{OPT}^2 \rfloor$. Therefore removing $\mathcal{H}$ increases the cost by at most $O(\epsilon)(\text{cost}(\text{OPT}) + \text{cost}(L))$, so that Step 1 is not too expensive.

We can therefore use this solution as a proxy for the optimal solution, and henceforth we will denote this solution by OPT. In particular, the badly cut facilities are defined for this solution and not the original OPT.

B. Structural Lemma

As in Section III, the algorithm computes a randomized hierarchical decomposition $\mathcal{D}$, and transforms the instance of the problem. Every badly cut client $c$ is moved to $L(c)$, namely, there is no more client at $c$ and we add an extra client at $L(c)$. Again, we let $\mathcal{I}_D$ denote the resulting instance and note that $\mathcal{I}_D$ is a random variable that depends on the randomness of $\mathcal{D}$.

Moreover, as for Facility Location, we let $B_D$ be the set of badly cut centers of $L$. We call $\text{cost}_2(S)$ the cost of a solution $S$ in the original instance $\mathcal{I}$, and $\text{cost}_2(S)$ its cost in $\mathcal{I}_D$. We let $\nu_{\mathcal{I}_D} = \max_{\text{solution } S} \text{cost}_2(S) - (1 + 3\epsilon)\text{cost}_2(S) - (1 - 3\epsilon)\text{cost}_2(S) - \text{cost}_2(S)$). We say that an instance $\mathcal{I}_D$ has small distortion (or $\nu_{\mathcal{I}_D} \leq \text{cost}_2(L)$ and there exists a solution $S$ that contains $B_D$ with $\text{cost}_2(S) \leq (1 + \epsilon)\text{cost}_2(\text{OPT}) + \epsilon\text{cost}_2(L)$. We go on with the next two steps of our construction, defining a solution $S^*$. We start with $S^* = \text{OPT}$ (and recall our convention OPT = OPT').

- **Step 2.** For each badly-cut facility $f \in L$ for which $\psi(f) \neq \emptyset$, let $f' \in \psi(f)$ be the closest to $f$. Replace $f'$ by $f$ in $S^*$.

- **Step 3.** Add all badly cut facility $f'$ of $L^0$ to $S^*$. We show next that $S^*$ satisfies the conditions for $\mathcal{I}_D$ to have small distortion with good probability.

**Lemma IV.2. The probability that $\mathcal{I}_D$ has small distortion is at least $1 - \epsilon$.**

**Proof:** The proof that $\nu_{\mathcal{I}_D} \leq \text{cost}_2(L)$ with probability at least $1 - \epsilon/2$ is identical to the one in Lemma III.1. We thus turn to bound the probability that solution $S^*$ satisfies the cardinality and cost requirements. Our goal is to show that this happens with probability at least $1 - \epsilon/2$. Then, taking a union bound over the probabilities of failure yields the proposition.

By definition, we have that $S^*$ contains $B_D$. We prove in the two following claims some properties on $S^*$.

**Claim IV.3. With probability at least $1 - \epsilon/4$, the set $S^*$ is an admissible solution, i.e., $|S^*| \leq k$.**

**Proof:** We let $b$ be the number of facilities of $L^0$ that are badly cut. By Lemma II.5, we have that $\mathbb{E}[b] \leq \epsilon^2 |L|/4$. By Markov’s inequality, the probability that $b$ is such that $b > \epsilon |L^0|/2$ is at most $\epsilon/2$. Now, condition on the event that $b \leq \epsilon |L^0|/2$. Since $|L^0| + |L^2| = |\text{OPT}^2|$, we have that $b \leq \epsilon |\text{OPT}^2|/2$. Moreover, the three steps converting OPT into $S^*$ ensure that $|S^*| \leq k + b - \epsilon |\text{OPT}^2|/2$. Combining the two inequalities gives $|S^*| \leq k$.

**Claim IV.4. With probability at least $1 - \epsilon/4$, cost$(S^*) \leq (1 + \epsilon(\epsilon)\text{cost}(\text{OPT}) + \epsilon \cdot \text{cost}(L))$**

**Proof:** We showed in Lemma IV.1 that the cost increase due to Step 1 is at most $O(\epsilon \cdot \text{cost}(\text{OPT}) + \text{cost}(L))$. We show now that Step 2 leads to a cost increase of $O(\epsilon \cdot \text{cost}(\text{OPT}) + \text{cost}(L))$ with good probability. For that, let $\text{OPT}_\text{close} := \{f \in \text{OPT} : f$ is the closest facility to $L(f)\}$. We show that the cost of replacing all $f \in \text{OPT}_\text{close}$ by $L(f) \in L$ is $O(\text{cost}(\text{OPT}) + \text{cost}(L))$. In order to prove that, we call the mixed solution the solution with facilities where every facility of $f \in \text{OPT}_\text{close}$ is replaced by $L(f)$. Note that $L(\text{OPT}_\text{close}) = L - L^0$.

For that, let $c$ be a client that is served in OPT by a facility $f$ of $\text{OPT}_\text{close}$. If $c$ is served in $L$ by a facility of $L - L^0$, then the facility appears in the mixed solution and the serving cost of $c$ is $\text{dist}(c, L)$. On the other hand, if $c$ is served by a facility $f_0$ of $L^0$ in $L$, then it is possible to serve it by the $L(f)$ that replaces $f$ in the mixed solution. The serving cost is therefore $\text{dist}(c, L(f)) \leq \text{dist}(c, f) + \text{dist}(L(f), L) \leq \text{dist}(c, f) + \text{dist}(L(f), f_0)$, using the definition of $L(f)$ for the last inequality. Using again the triangle inequality, this cost is at most $2\text{dist}(c, f) + \text{dist}(f, f_0)$. Moreover, any client served by a facility of $\text{OPT} - \text{OPT}_\text{close}$ is served by its optimal facility in the mixed solution, with cost $\text{dist}(c, \text{OPT})$. Hence the cost of the mixed solution is at most $2\text{cost}(\text{OPT}) + \text{cost}(L)$.

Moreover, the probability of replacing $f \in \text{OPT}_\text{close}$ by $L(f) \in L - L^0$ in Step 2 is the probability that $L(f)$ is badly cut, which is $\Theta(\epsilon, p)$ by Lemma II.5. Finally, with linearity of expectation, the expected cost to add the badly cut facilities $f \in L - L^0$ instead of their closest facility of OPT in Step 2 is $O(\epsilon(\epsilon)\text{cost}(\text{OPT}) + \text{cost}(L))$. Markov’s inequality thus implies that the cost of the first step is at most $O(\epsilon(\epsilon)\text{cost}(\text{OPT}) + \text{cost}(L))$ with probability $1 - \frac{\epsilon}{O(\epsilon(\epsilon))} = 1 - \epsilon/4$, since $\epsilon(\epsilon) \leq \epsilon^2/4$ in the case of $k$-Median.

**Lemma IV.2** follows from taking a union bound over the probabilities of failure of Claim IV.3 and IV.4.

Condition now on $\mathcal{I}_D$ having small distortion, and let OPT' be the solution containing $B_D$ with cost $(1 + \epsilon)\text{cost}_2(\text{OPT}) + \epsilon\text{cost}_2(L)$. We have to prove the same structural lemma as for Facility Location, to say that there exists
a portal-respecting solution with cost close to cost(OPT).

Recall that $L_c$ and OPT are the distances from the original position of $c$ to $L$ and OPT; but $c$ may have been moved to $L(c)$. Recall also that OPT is defined after removing some centers in Step 1.

**Lemma IV.5.** Condition on $\mathcal{D}$ having small distortion. For any client $c$ in $\mathcal{D}$, let OPT' be the closest facility to $c$ in OPT. Then $c$ and OPT' are separated in $\mathcal{D}$ at level at most $\log(7(L_c + OPT_c)) + \tau(\varepsilon, d)$.

**Proof:** The proof of this lemma is very similar to the one of Lemma III.2. However, since some facilities of OPT were removed in Step 2, we need to adapt the proof carefully.

Let $c$ be a client. If OPT(c) was removed in Step 2, it was replaced by a facility $f$ such that $dist(OPT(c), f) \leq dist(OPT(c), L(c))$ (because $L(OPT(c)) = f$ means that $f$ is the facility of $L$ closest to OPT(c)). Therefore

$$dist(c, f) \leq 2dist(c, OPT(c)) + dist(c, L(c)).$$

(1)

To find the level at which $c$ and OPT' are separated, we distinguish between two cases: either $c$ is badly cut, or not.

If $c$ is badly cut, then it is now located at $L(c)$ in the instance $\mathcal{D}$. In that case, either:

1) $L(c)$ is also badly cut, and therefore $L(c) \in$ OPT and so OPT' = $L(c)$. It follows that $c$ and OPT' are never separated.

2) $L(c)$ is not badly cut. Then $dist(c, OPT'(c)) = dist(L(c), OPT'(L(c))).$ OPT' is not necessarily in OPT; in that case, it was replaced by a facility $f$ that verifies $dist(c, f) \leq 2dist(c, OPT(c)) + dist(c, L(c))$, by Property (1). Since $dist(c, L(c)) = 0$, we have (either if OPT(L(c)) \in OPT' or not) that $dist(c, OPT'(c)) \leq 2OPT'(c)$.

Since $L(c)$ is not badly cut, the ball $\beta(L(c), 2OPT(L(c)))$ is cut at level at most $\log(4OPT(L(c)) + \tau(\varepsilon, d))$. By triangle inequality, $OPT(L(c)) = dist(L(c), OPT'(L(c))) \leq L_c + OPT_c$, and thus $c$ and OPT' are also separated at level at most $\log(4L_c + 4OPT_c) + \tau(\varepsilon, d)$.

In the other case, $c$ is not badly cut, all of the balls $\beta(c, 2')$ where $2' \in [\varepsilon L_c, L_c/\varepsilon]$ are not badly cut, and $c$ is not moved to $L(c)$. We make a case distinction according to OPT_c.

1) If $L_c \leq \varepsilon OPT_c$, then we have the following. If $L(c)$ is badly cut, $L(c)$ is open and therefore OPT'(c) = $L_c$. Moreover, since $c$ is not badly cut the ball $\beta(c, L_c)$ is cut at level at most $\log L_c + \tau(\varepsilon, d)$. Therefore $c$ and OPT' are separated at level at most $\log L_c + \tau(\varepsilon, d)$. In the case where $L(c)$ is not badly cut, both $c$ and OPT' lie in the ring centered at $L(c)$ and of diameter $3OPT(L(c))$. Indeed,

$$dist(c, L(c)) \leq \varepsilon dist(c, OPT(c)) \leq \varepsilon dist(c, OPT(L(c))) \leq \varepsilon dist(c, L(c)) + \varepsilon(\varepsilon, d).$$

And therefore, for any $\varepsilon \leq 3/4$,

$$dist(c, L(c)) \leq \varepsilon OPT_c \leq 3OPT(L(c)).$$

which is smaller than $3OPT(L(c)$ for any $\varepsilon \leq 1/2$. Hence we have $c,OPT(c) \in \beta(L(c), 3OPT(L(c))).$ To apply the definition of badly cut, we need to consider rings with radius power of 2: let us therefore pick $i$ such that $3OPT(L(c)) \in (2^{i-1}, 2^i)$ (note that $2^i \leq 6OPT(L(c))$). Since $L(c)$ is not badly cut, this ring is not badly cut either and thus $c$ and OPT' are separated at level at most $\varepsilon$.

2) If $L_c \geq OPT_c/\varepsilon$ then, since $c$ is not badly cut, the ball centered at $c$ with radius $\varepsilon L_c$ is not badly cut. Since we have $dist(c, OPT'(c)) \leq 2OPT_c + L_c \leq 2L_c$, $c$ and OPT' lie in the ball $\beta(c, 2L_c)$ and are thus cut at level at most $\log(4L_c) + \tau(\varepsilon, d)$. Moreover, OPT'(c) lies in this ball, which concludes the lemma.

Equipped with these two lemmas, we can prove the following lemma, which concludes the section:

**Lemma IV.6.** Condition on $\mathcal{D}$ having small distortion. There exists a portal-respecting solution $S$ in $\mathcal{D}$ is such that $cost_1(S) + B(S) \leq (1 + O(\varepsilon))cost_1(OPT) + O(\varepsilon)cost_2(L))$.

**Proof:** The proof follows exactly the one of Lemma III.3, using the definition of small distortion, OPT', Lemma IV.2, and Lemma IV.5.

**Extension to k-Means:** The adaptation to k-Means can be essentially captured by the following inequality: $(x + y)^2 \leq 2(x^2 + y^2)$. Indeed, taking the example of Claim IV.4, the detour $dist(c, f') \leq 3dist(c, f) + 2dist(c, l)$ gives a cost $dist(c, f')^2 = O(dist(c, f)^2 + dist(c, l)^2)^2 + dist(c, f') \cdot dist(c, f) = O(dist(c, f)^2 + dist(c, l)^2)$. This follows through all the other lemmas, and therefore the
structural lemma holds also for $k$-Means.

C. The Algorithm

The algorithm follows the lines of the one for Facility Location, in Section III-C. It first computes a constant-factor approximation $L$, then the hierarchical decomposition $D$ (with parameter $\rho = \epsilon^{-1} + d$) and constructs instance $I_D$. A dynamic program is then used to solve efficiently the problem, providing a solution $S$ of cost at most $(1 + \epsilon) \text{cost}_I(\text{OPT})$ – conditioned on the event that the instance $I_D$ has small distortion.

Dynamic programming: The algorithm proceeds bottom up along the levels of the decomposition. We give an overview of the dynamic program which is a slightly refined version of the one presented for Facility Location in Section III-C. We make use of two additional ideas.

To avoid the dependency on $k$ we proceed as follows. In the standard approach, a cell of the dynamic program is defined by a part of the decomposition $D$, the portal parameters (as defined in Section III-C), and a value $k_0 \in [k]$. The value of an entry in the table is then the cost of the best solution that uses $k_0$ centers, given the portal parameters.

For our dynamic program for the $k$-Median and $k$-Means problems, we define a cell of the dynamic program by a part $B$, the portal parameters $(t_1, \ldots, t_{n_B})$ and $(s_1, \ldots, s_{n_B})$ and a value $c_0$ in $[\text{cost}(L)/n; (1 + \epsilon) \text{cost}(L)]$. The entry of the cell is equal to the minimum number $k_0$ of centers that need to be placed in part $B$ in order to achieve cost at most $c_0$, given the portal parameters. Moreover, we only consider values for $c_0$ that are powers of $(1 + \epsilon / \log n)$. The output of the algorithm is the minimum value $c_0$ such that the root cell has value at most $k$ (i.e., the minimum value such that at most $k$ centers are needed to achieve it).

The DP table can be computed the following way. For the parts that have no descendant, namely the base cases, computing the best clustering given a set of parameters can be done easily: there is at most one client in the part, and verifying that the parameter values for the centers inside the part are consistent can be done easily. At a higher level of the decomposition, a solution is obtained by going over all the sets of parameter values for all the children parts. It is immediate to see whether sets of parameter values for the children can lead to a consistent solution (similar to [25], [4]). Since there are at most $2^O(d)$ children parts, this gives a running time of $O^*(2^O(d))$, where $q$ is the total number of parameter values.

This strategy would lead to a running time of $f(c, d)n \log^{O(d)} n$. We can however treat the children in order, instead of naively testing all parameter values for them. We use a classical transformation of the dynamic program, in which the first table is filled using an auxiliary dynamic program. A cell of this auxiliary DP is a value $c_0$ in $[\text{cost}(L)/n; (1 + \epsilon) \text{cost}(L)]$, a part $C$, one of its children $C_1$, and the portal parameters for the portals of $C$ and all its children before $C_1$ in the given order. The entry of the cell is equal to the minimum number of centers $k_0$ that need to be placed in the children parts following $C_1$ to achieve a cost of $c_0$ given the portal parameters. To fill this table, one can try all possible sets of parameters for the following children, see whether they can lead to a consistent solution, and compute the minimum value among them.

Analysis – proof of Theorem I.1 and Theorem I.2.: We first show that the solution computed by the algorithm gives a $(1 + (1 + \epsilon) \log n)$-approximation, and then prove the claim on the complexity.

Lemma IV.7. Let $S^*$ be the solution computed by the algorithm. With probability $1 - \epsilon$, it holds that $\text{cost}_I(S^*) = (1 + O(\epsilon)) \text{cost}_I(\text{OPT}) + O(\epsilon \text{cost}_I(L))$.

Proof: With probability $1 - \epsilon$, $I_D$ has small distortion (Lemma IV.2). Following Lemma IV.6, let $S$ be a portal-respecting solution such that $\text{cost}_I(S) + B(S) \leq (1 + O(\epsilon)) \text{cost}_I(\text{OPT}) + O(\epsilon \text{cost}_I(L))$.

As in Lemma III.4, $S$ can be adapted to a configuration of the DP with a small extra cost. The cost incurred to the rounding of distances can be charged either to $B(S)$ or is a $O(\epsilon \text{cost}_I(S))$, as in Lemma III.4. The cost to round the value $c_0$ is a $(1 + \epsilon / \log n)$ factor at every level of the decomposition. Since there are $O(\log n)$ of them, the total factor is $(1 + \epsilon / \log n)^{O(\log n)} = 1 + O(\epsilon)$. Hence, we have the following:

$$\text{cost}_I(S^*) = (1 + O(\epsilon)) \text{cost}_I(S) + (O(\epsilon \text{cost}_I(L))$$

(Since $I_D$ has small distortion)

$$= (1 + O(\epsilon))((\text{cost}_I(S) + B(S))$$

(Following the previous paragraph)

$$\leq (1 + O(\epsilon) \text{cost}_I(\text{OPT}) + O(\epsilon \text{cost}_I(L))$$

(By definition of $S$)

Lemma IV.8. The running time of the DP is $n \cdot (1/\epsilon)^{O(\log n)} \cdot \log^4 n$.

Proof: The number of cells in the auxiliary DP is given by the number of parts $(O(n))$, the number of children of a part $(2^{O(d)})$, the number of portal parameters $((1/\epsilon)^{O(d)}/\epsilon)$ and the possible values for $c_0 (O(\log^2 n))$: it is therefore $n \cdot 2^{O(d)} \cdot (1/\epsilon)^{O(d)} / \epsilon \cdot \log^4 n$.

The complexity to fill the table adds a factor $(1/\epsilon)^{O(d)} / \epsilon \cdot \log^2 n$, to try all possible combination of portal parameters and value of $c_0$. Hence, the overall running time of the DP is $n \cdot (1/\epsilon)^{O(d)} / \epsilon \cdot \log^4 n$.

The proof of Theorem I.1 and Theorem I.2 are completed by the following lemma, which bounds the running time of the preprocessing steps.

553
Lemma IV.9. For k-Median and k-Means, the total running time of the algorithms are respectively $2O(d)n \log^9 n + f(\varepsilon, d)n \log^{10} n$ and $2O(d)n \log^{10} n + f(\varepsilon, d)n \log^5 n$, where $f(\varepsilon, d) = \left(\frac{1}{\varepsilon}\right)^{O(\varepsilon^{-2})}/\varepsilon$.

Proof: We need to bound the running time of three steps: computing an approximation, computing the hierarchical decomposition, and running the dynamic program.

For k-Median, a constant-factor approximation can be computed in $O(n \log^9 n) = 2O(d)n \log^9 n$ time with Thorup’s algorithm [38]. The split-tree decomposition can be found in $2O(d)n \log^9 n$ time as explained in Section II. Moreover, as explained in Lemma IV.8, the dynamic program runs in time $f(\varepsilon, d)n \log^4 n$, ending the proof of the Theorem I.1.

Another step is required for k-Means. It is indeed not known how to find a constant-factor approximation in near-linear time. However, one can notice that a $c$-approximation for k-Median is an $nc$-approximation for k-Means, using the Cauchy-Schwarz inequality. Moreover, notice that starting from a solution $S$, our algorithm finds a solution with cost $(1 + O(\varepsilon))\text{cost}(OPT) + O(\varepsilon)\text{cost}(S)$ in time $f(\varepsilon, d)n \log^4 n$, as for k-Median.

Repeating this algorithm $N$ times, using in step $i + 1$ the solution given at step $i$, gives thus a solution of cost $(1 + O(\varepsilon))\text{cost}(OPT) + O(\varepsilon^N)\text{cost}(S)$. Starting with cost(\text{cost}) = $O(n)\text{cost}(OPT)$ and taking $N = O(\log n)$ ensures to find a solution for k-Means with cost $(1 + O(\varepsilon))\text{cost}(OPT)$. The complexity for k-Means is therefore the same as for k-Median, with an additional $\log n$ factor. This concludes the proof of Theorem I.2.

V. Other Applications of the Framework

Our techniques can be generalized to variants of the clustering problems where outliers are taken into account. We consider here two of them: k-Median with Outliers and its Lagrangian relaxation, Prize-Collecting k-Median. It can also be used to find a bicriteria approximation to k-Center.

A. Prize-Collecting k-Median

In the “prize-collecting” version of the problems, it is possible not to serve a client $c$ by paying a penalty $p_c$ (these problems are also called clustering “with penalties”). For a solution $S$, we call an outlier for $S$ a client that is not served by $S$. Formally, an instance is a quintuple $(C, F, \text{dist}, p, k)$ where $(C \cup F, \text{dist})$ is a metric, $k$ is an integer and $p: C \rightarrow \mathbb{R}^+$ the penalty function, and the goal is to find $S \subseteq F$ and $O \subseteq C$ such that $|S| = k$ and $\sum_{c \in C \setminus O} \text{dist}(c, F) + \sum_{c \in O} p_c$ is minimized.

Looking at the Prize-Collecting k-Median problem, we aim at applying the framework from Section IV. Let $L = (L_C, L_O)$ be an approximate solution: $L_C$ is a set of centers, $L_O$ a set of outliers. We define badly cut for outliers as we did for centers: an outlier $c$ of $L_O$ is badly cut w.r.t. $D$ if there exists an integer $i$ such that $2^i \in [\varepsilon OPT_c, OPT_c/\varepsilon]$ and the ball $\beta(v, 2^i)$ is cut at some level $j$ greater than $i + \tau(\varepsilon, d)$, where OPT is the distance from $c$ to the closest facility of the optimum solution OPT. Hence, Lemma II.5 extends directly, and the probability that an outlier in $L_O$ is badly cut is $e(\varepsilon, p)$.

We now turn to the previous framework, showing how to construct a near-optimal solution containing all badly cut centers of $L$. For that we transfer the definitions of the mappings $L_C, \phi$ ($L_C$ maps a client to its closest center of $L$, and $\phi(\ell) = \{f_{\ell} \in \text{OPT} \mid L(f_{\ell}) = 1\}$) and of the sets $L^0, L^1, L_{\geq 2}, \text{OPT}^1, \text{OPT}_{\geq 2}$. We will show that this framework, with only a few modifications, leads to an approximation scheme for Prize-Collecting k-Median. Let $S^\ast = \text{OPT}$. As in Section IV, we start by removing a few centers from the optimal solution, without increasing the cost too much:

- **Step 1.** Among the facilities of OPT_{\geq 2} that are not the closest of their corresponding facility in $L_{\geq 2}$, remove from $S^\ast$ the subset $\mathcal{H}$ of size $|\varepsilon(\text{OPT}_{\geq 2})/2|$ that yields the smallest cost increase, i.e., the smallest value of $\sum_{c \in C \setminus L_O} d(c, S^\ast - \mathcal{H}) + \sum_{c \in L_O} \varepsilon \text{cost}(c) \in \mathcal{H} \cdot p_c$.

The function minimized by $\mathcal{H}$ corresponds to redirecting all clients served in the local solution to a center of $S^\ast - \mathcal{H}$ and paying the penalty for clients $c \in L_O$ such that OPT($c$) $\in \mathcal{H}$. Those clients are thus considered as outliers in the constructed solution.

Lemma V.1. After step 1, $S^\ast$ has cost $(1 + O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon)\text{cost}(L)$.

Proof sketch: The proof is essentially the same as Lemma IV.1, with an averaging argument: for a client $c$, the cost of removing OPT($c$) from $S^\ast$ is $O(\text{OPT}_c + L_c)$: if $c \notin L_O$, the argument is the same as in Lemma IV.1, and if $c \in L_O$ the cost is $p_c = L_c$. Hence the proof follows.

Again, we denote now by OPT this solution $S^\ast$ and define the instance $I_D$ according to this solution. Recall that $B_D$ is the set of badly cut centers of $L$, and denote $O_D$ the set of badly cut outliers of $L$. We say that an instance $I_D$ has small distortion if $\text{dist}_{c, \ell} \leq \varepsilon \text{cost}(L)$ and there exists a solution $S$ that contains $B_D$ as centers and $O_D$ as outliers with cost cost$_\ell(S) \leq (1 + \varepsilon) \text{cost}_\ell(\text{OPT}) + \varepsilon \text{cost}_\ell(L)$.

To deal with the badly cut centers, there is only one hurdle to be able to apply the proof of Lemma IV.6. Indeed, when the algorithm deletes a center of OPT that serves a client $c$, it is possible to bound the cost of reassigning $c$ using dist($c, S$). However this is not possible to do when $c$ is an outlier for $S$: there is no control on the cost dist($c, S$), and hence one has to pay the penalty $p_c$. It is thus necessary to find a mechanism that ensures to pay this penalty only with a probability $\varepsilon$ for each client $c$. Similar to Section IV, this is achieved with the following three steps:
• Step 2. For each badly-cut facility \( f \in L \) for which \( \psi(f) \neq \emptyset \), let \( f' \in \psi(f) \) be the closest to \( f \). Replace \( f' \) by \( f \) in \( S^* \). For all clients \( c \in L_O \) such that \( \text{OPT}(c) = f' \), add \( c \) as outliers.

• Step 3. Add all badly cut facility \( f' \) of \( L' \) to \( S^* \).

• Step 4. Add all badly cut outliers of \( L \) to the outliers of \( S^* \).

We show next that \( S^* \) satisfies the conditions for \( \mathcal{I}_D \) to have small distortion with good probability.

**Lemma V.2.** The probability that \( \mathcal{I}_D \) has small distortion is at least \( 1 - \varepsilon \).

**Proof:** When bounding the cost increase due to Step 2, it is necessary to add as outliers all clients served by \( f' \) that are outliers in \( L \). Since \( f' \) is deleted from \( S^* \) with probability \( \kappa(\varepsilon, p) \), the expected cost due to this is \( \sum_{c \in L_O} \kappa(\varepsilon, p) \cdot p_c \leq \kappa(\varepsilon, p) \text{cost}_D(L) \). Using Markov’s inequality, this is at most \( \varepsilon / \text{cost}_D(L) \) with probability \( 1 - \varepsilon / 3 \).

Step 3 does not involve outliers at all. Hence, Claim IV.3 and IV.4 are still valid. Combined with the previous observation about Step 2, this proves that after Step 3, \( S^* \) contains at most \( K \) centers - including the ones in \( B_D \) - and has cost at most \( (1 + \varepsilon) \text{cost}_D(\text{OPT}) + \varepsilon / \text{cost}_D(L) \) with probability at least \( 1 - \varepsilon / 3 \).

Step 4 implies that all outliers in \( O_D \) are also outliers in the constructed solution. Moreover, since an outlier of \( L \) is badly cut with probability \( \kappa(\varepsilon, p) \), the expected cost increase due to this step is at most \( \kappa(\varepsilon, p) \text{cost}_D(L) \). Using again Markov’s inequality, this cost is at most \( \varepsilon / \text{cost}_D(L) \) with probability \( 1 - \varepsilon / 3 \).

By union-bound, the solution \( S^* \) has cost at most \( (1 + \varepsilon) \text{cost}_D(\text{OPT}) + \varepsilon / \text{cost}_D(L) \) with probability \( 1 - \varepsilon \). Hence, \( \mathcal{I}_D \) has small distortion with probability \( 1 - \varepsilon \).

Given an instance with low distortion, it is again possible to prove that there exists a near optimal portal-respecting solution, and the same DP as for \( k \)-Median can find it.

Therefore, using the polynomial time algorithm of Charikar et al. [10] to compute a constant-factor approximation, the algorithm presented in Section IV can be straightforwardly adapted, concluding the proof of Theorem I.4.

**B. k-Median with Outliers**

In the \( k \)-Median with Outliers problem, the number of outliers allowed is bounded by some given integer \( z \). We do not manage to respect this bound together with having at most \( k \) facilities and a near-optimal solution: we need to relax it a little bit, and achieve a bicriteria approximation, with \( k \) facilities and \( (1 + O(\varepsilon))z \) outliers. For this, our framework applies nearly without a change.

The first step in the previous construction does not apply directly: the “cost” of removing a center is not well defined. In order to fix this part, Step 1 is randomized: among the facilities of \( \text{OPT}^{2^2} \) that are not the closest of their corresponding facility in \( L^{2^2} \), remove from \( S^* \) a random subset \( \mathcal{H} \) of size \( \varepsilon \cdot |\text{OPT}^{2^2}| / 2 \).

**Lemma V.3.** After the randomized Step 1, \( S^* \) has expected cost \( (1 + O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon)\text{cost}(L) \).

**Proof:** Since there are at least \( |\text{OPT}^{2^2}| / 2 \) facilities of \( \text{OPT}^{2^2} \) that are not the closest of their corresponding facility in \( L^{2^2} \), the probability to remove one of them is \( O(\varepsilon) \).

Hence, every outlier of \( L \) that is served in \( \text{OPT} \) must be added as an outlier in \( S^* \) with probability \( O(\varepsilon) \) – when its serving center in \( \text{OPT} \) is deleted. Hence, the expected number of outliers added is \( O(\varepsilon z) \).

Moreover, the proof of Lemma IV.1 shows that the sum of the cost of deleting all possible facilities is at most \( O(\text{cost}(\text{OPT}) + \text{cost}(L)) \) (adding a point as outlier whenever it is necessary). Removing each one of them with probability \( O(\varepsilon) \) ensures that the expected cost of \( S^* \) after step 1 is \( (1 + O(\varepsilon))\text{cost}(\text{OPT}) + O(\varepsilon)\text{cost}(L) \).

The three following steps are the same as in the previous section, and the proof follows: with constant probability, the instance \( \mathcal{I}_D \) has small distortion (defined as for \( k \)-Median with penalties), and one can use a dynamic program to solve the problem on it. The DP is very similar to the one for \( k \)-Median. The only difference is the addition of a number \( x \) to each table entry, which is a power of \( (1 + \varepsilon / \log n) \), and represents the (rounded) number of outliers allowed in the subproblem. This adds a factor \( \log^2 n / \varepsilon \) to the complexity.

It is possible to compute a constant factor approximation \( S \) in polynomial time (using Krishnasawamy et al. [26]). Hence, this algorithm is a polynomial time bicriteria approximation scheme for \( k \)-Median with outliers. As in Section IV, this directly extends to \( k \)-Means with outliers.

This concludes the proof of Theorem I.5.

**C. k-Center**

In the \( k \)-Center problem, the goal is to place \( k \) centers such as to minimize the largest distance from a point to its serving center. We propose a bicriteria approximation, allowing the algorithm to open \( (1 + O(\varepsilon))k \) centers.

For this, we change slightly the definition of badly-cut. Given a solution \( L \) with cost \( \gamma \) and a hierarchical decomposition \( D \), a center \( f \) of \( L \) is badly cut w.r.t \( D \) if the ball \( \beta(f, 2^i) \) is cut at some level \( j \) greater than \( i + \tau(\varepsilon, d) \), for \( i \) such that \( 2^{i-1} \leq 2\gamma \leq 2^i \).

Note that Lemma II.5 still holds with this definition: a center \( f \) is badly cut with probability at most \( \kappa(\varepsilon, p) \). Let \( B_D \) be the set of badly cut centers. We assume in the following that \( L \) is a 2-approximation, i.e. \( \gamma \leq 2 OPT \).

We make the crucial following observation, using the doubling property of the metric. Let \( f \) be a center of \( L \). By definition of doubling dimension, the ball \( \beta(f, \gamma) \) can be covered by \( 2^d \) balls of radius \( \gamma / 2 \leq \text{OPT} \). Let \( C_c \) be the set of centers of such balls, such that \( \beta(f, \gamma) \subseteq \bigcup_{f' \in C_c} \beta(f', \gamma / 2) \).

555
Given an instance $I$, we construct $I_D$ the following way: for each badly cut facility $f$, open all the facilities in $C_f$, and remove all the clients in $\beta(f,\gamma)$ from the instance. We let $C = \bigcup_f C_f$. The structural lemma of this section is the following:

**Lemma V.4.** It holds that for all solution $S$:

- $\text{cost}_{\pi}(S) \leq \text{cost}_{\pi}(S, \text{OPT})$
- $\text{cost}_{\pi}(S \cup C) \leq \max(\text{cost}_{\pi}(S), \text{OPT})$

**Proof:** Since the instance $I_D$ contains a subset of clients of $I$, it holds that $\text{cost}_{\pi}(S) \leq \text{cost}_{\pi}(S)$. Let $S$ be a solution in $I_D$. It serves all client in $I$ but the one removed: these ones are served by $C$ at a cost $\gamma/2 \leq \text{OPT}$. Hence, the cost of $S \cup C$ is at most $\max(\text{cost}_{\pi}(S), \text{OPT})$.

We now show, in a similar fashion as Lemma III.2, that the clients in $I_D$ are cut from their serving facility of OPT at a controlled level. Recall that OPT is defined for instance $I$.

**Lemma V.5.** Let $c$ be a client in $I_D$ and $\text{OPT}(c)$ its serving facility in OPT. $C$ and $\text{OPT}(c)$ are cut at level at most $\log(2\gamma) + \tau(\varepsilon, d)$.

**Proof:** Let $c$ be a client, $L(c)$ its serving center in $L$ and $\text{OPT}(c)$ its serving center in OPT. If $c$ is still a client in $I_D$, it means that $L(c)$ is not badly cut. Observe that $\text{dist}(L(c), \text{OPT}(c)) \leq \text{dist}(c, L(c)) + \text{dist}(c, \text{OPT}(c)) \leq \gamma + \text{OPT} \leq 2\gamma$

Let $i$ such that $2^{i-1} \leq 2\gamma \leq 2^i$. Since $L(c)$ is not badly cut, the ball $\beta(L(c), 2^i)$ is not badly cut neither: hence, $c$ and $\text{OPT}(c)$ (that are in this ball) are cut at level at most $i + \tau(\varepsilon, d) \leq \log(2\gamma) + \tau(\varepsilon, d)$.

This lemma is stronger than Lemmas III.2 and IV.5: it allows us to consider only levels of the decomposition with diameter less than $2^{i+\tau(\varepsilon, d)}$.

Since the set $C$ has expected size $\kappa(\varepsilon, p)k$, Markov’s inequality ensures that with probability $1 - \varepsilon$ this set has size $O(\varepsilon)k$. If every part with diameter $\Delta$ of the hierarchical decomposition is equipped with a $\rho\Delta$-net (for $\rho = \varepsilon^{2-\tau(\varepsilon, d)}$), Lemma V.5 ensure that there exists a portal-respecting solution $S$ with cost $\text{cost}_{\pi}(S) \leq \text{OPT} + O(\varepsilon)\gamma = (1 + O(\varepsilon))\text{OPT}$. Lemma V.4 ensures that lifting this solution back to $I$ and adding $C$ as centers gives a near-optimal solution.

Using the same algorithm as for $k$-Medians to compute a good portal-respecting solution, and computing a 2-approximation with a simple greedy algorithm (see e.g. [17]), that runs in time $O(n \log k)$ concludes the proof of Theorem I.6.

VI. ACKNOWLEDGMENT

Andreas Emil Feldman is supported by the Czech Science Foundation GAČR (grant #17-10090Y), and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004).


REFERENCES


APPENDIX

A. Proof of Section II

Proof of Lemma II.3: We present the algorithm constructing the hierarchical decomposition and proves the lemma as a second step.
Without loss of generality, assume that the smallest distance in the metric is 1: the aspect-ratio $\Delta$ is therefore the diameter of the metric. Start from a hierarchy of nets $Y_0 := V, \ldots, V_{\log(\Delta)}$ such that $Y_i$ is a $2^{i-2}$-net of $Y_{i-1}$. Moreover, pick a random order on the points $V$ and a random number $r \in [1/2, 1)$. The hierarchical decomposition $D$ is defined inductively, starting from $V_{\log(\Delta)} = V$. To partition a part $B$ at level $i$ into subpart at level $i - 1$, do the following: for each $y \in Y_{i-1} \cap B$ taken in the random order, define $B \cap \beta(y, 2r^2)$ to be a part at level $i - 1$ and remove $B \cap \beta(y, 2r^2)$ from $C$.

When we assume access to the distances through an oracle, it is possible to construct this hierarchy an augment it with the set of portals in time $(1/\rho)^{O(\log n)} \log(\Delta)$. Moreover, these portals can be made nested, meaning that portals at level $i + 1$ are also portals at level $i$ [22], [15].

We prove now that this hierarchical decomposition has the required properties. The diameter of each part is bounded by $2^{i+1}$ by construction; therefore to have Property 2 it is enough to make $P$, an $(\varepsilon/2^{i+1})$-net of $V$. The Lemma II.1 ensures the conciseness, and the definition of a net ensures that every point is at distance $\varepsilon 2^{i+1}$ of $P$, which implies the preciseness. Proving the scaling property requires a bit more work.

The two ingredients needed for this part stem from the construction of the decomposition: the diameter of any part at level $i$ is at most $2^{i+1}$, and the minimum distance between two points of $V_i$ is bigger than $2^{i-2}$.

These two properties are enough in order to prove our lemma. Let $i$ be a level such that $2^i \leq r$, then $r/2^i = \Omega(1)$ so there is nothing to prove. Otherwise, we proceed in two steps. First, let us count the number of level $i$ parts that could possibly cut a ball $\beta(x, r)$. A level $i$ part is included in a ball $\beta(y, 2r^2)$ for some $y \in Y_i$; therefore if $\text{dist}(x, y) > r + 2r^2$ then $y$’s part cannot cut $\beta(x, r)$. So it is required that $\text{dist}(x, y) \leq r + 2r^2 \leq 2r^2$. But since the minimum distance between two points of $Y_i$ is $2^{i-2}$, and $Y_i$ has doubling dimension $d$, we have $|Y_i \cap \beta(x, 2r^2)| = 2^{d \log(2^{i/2^2})} = 2d^2$. Thus there is only a bounded number of parts to consider.

We prove for each of them that the probability that it cuts $\beta(x, r)$ is $O(r/2^i)$. A union-bound on all the possible parts is then enough to conclude. Let therefore $y \in Y_i \cap \beta(x, 2r^2)$, and $x_m$ and $x_M$ be the respective closest and farthest point of $\beta(x, r)$ from $y$. A necessary condition for $y$’s part to cut $\beta(x, r)$ is that the diameter of the part is in the open interval $[d(y, x_m), d(y, x_M)]$. Since $x_m, x_M \in \beta(x, r)$ this interval has size $2r$, and the radius of the part is picked uniformly in $[2r/2, 2r^2]$. Therefore the probability that the radius of the part falls in $[d(y, x_m), d(y, x_M)]$ is at most $4r/2^i$. And finally, the probability that $y$’s part cuts $\beta(x, r)$ is indeed $4r/2^i$.

By a union-bound over all the parts that could possibly cut $\beta(x, r)$ we obtain the claimed probability $\Pr[C \text{ cuts } \beta(x, r) \text{ at a level } i] = 2d^2 r^2/2^i$.

Lemma A.1. Let $P$ be a problem among Facility Location, $k$-Medians or $k$-Means. Given an instance $(I, \text{dist})$ with $n$ points, $\varepsilon > 0$ and a constant-factor approximation for $P$ on $I$, there exists a linear-time algorithm that outputs a set of instances $(I_1, \text{dist}_1), \ldots, (I_m, \text{dist}_m)$ such that

- $I_1, \ldots, I_m$ is a partition of $I$;
- for all $i$, $I_i$ has aspect-ratio $O(n^i/\varepsilon)$;
- if $(\bigcup I_i, \min \text{dist})$ is the instance where distances between points of the same part $I_i$ are given by dist, and distances between points of different parts is set to $\infty$, then
  - there exists a solution on $\bigcup I_i$ with cost $(1 + \varepsilon/n) \text{cost}(OPT)$, and
  - any solution on $\bigcup I_i$ of cost $X$ induces a solution of cost at most $X + \text{cost}(OPT)/n$ on $I$.

Proof: The cost of the constant-factor approximation is an estimate $\gamma$ on the cost of the optimum solution $OPT$: $\gamma = \Theta(\text{cost}(OPT))$. It is then possible to replace all distances longer than $2\gamma$ by $\infty$: distances longer than $\gamma$ will indeed never be used by solution with cost better than $\gamma$, so the cost of these solutions is preserved after this transformation. We say that two vertices are connected if their distance is not $\infty$, and call a connected component any maximal set of connected vertices. The transformation ensured that any connected component has diameter at most $2OPT$, and that every cluster of OPT is contained inside a single connected component. Moreover, any connected component has doubling dimension $2d$: indeed, a subspace of a metric with doubling dimension $d$ has a doubling dimension at most $2d$. Note also that this transformation can be made implicitly: every time the algorithm queries an edge, it can replace the result by $\infty$ if necessary.

To identify the connected component, the algorithm builds a spanner with the algorithm of [22]: the connected components of the spanner are exactly the ones of our metric, and can be found in linear time.

Then, for each connected component, the algorithm defines an instance of the more general version of the clustering problem by the following way. It first sets $\chi(v) = 1$ for all vertex $v$. Then, it iterates over all edges, it contracts every edge $(u, v)$ with length less than $(\varepsilon \cdot \gamma/n^3)$ to form a new vertex $w$ and sets $\chi(v) = \chi(u) + \chi(v)$.

Now, we aim at reconstructing a metric from this graph. We will do it in an approximate way: for all connected points $u, v$ of connected component $i$, we set $\text{dist}_i(u, v) = 0$ if $u$ and $v$ are merged in the graph, and otherwise $\text{dist}(u, v)$. This ensures that $\varepsilon \cdot \gamma/n^3 \leq \text{dist}_i(u, v) \leq 2\gamma$, hence the aspect-ratio of $I_i$ is $O(n^i/\varepsilon)$. Moreover, every distance is preserved up to an additive $O(\varepsilon \cdot \text{cost}(OPT)/n^2)$. Since every cluster of OPT is contained inside a single connected component, this ensures that OPT induces a solution of cost $(1 + \varepsilon/n)\text{cost}(OPT)$ on $\bigcup I_i$. Moreover,
lifting a solution in $\bigcup \mathcal{I}_i$ to $\mathcal{I}$ costs at most $\varepsilon \text{cost}(\text{OPT})/n^2$ per pair (client, center) and therefore $\varepsilon \text{cost}(\text{OPT})/n$ in total.

If the problem considered is Facility Location, it is easy to merge the solutions on subinstances: since there is no cardinality constraint, the global solution is simply the union of all the solutions. The hard constraint on $k$ makes things a bit harder. Note that the dynamic program presented in Section IV naturally handles it without any increase in its complexity; however, for completeness we present now a direct reduction.

**Lemma A.2.** Given a problem $P$ among $k$-Medians or $k$-Means, a set of instances $(\mathcal{I}_1, \text{dist}_1), \ldots, (\mathcal{I}_m, \text{dist}_m)$ given by Lemma A.1 and an algorithm running in time $n_i(\log n_i)^{\alpha t}(\Delta)$ to solve $P$ on instances with $n_i$ points and aspect-ratio $\Delta$, there exists an algorithm that runs in time $O(n(\log n)^{\alpha+2}(O(n^4/\varepsilon)))$ to solve $P$ on $\bigcup \mathcal{I}_i$.

**Proof:** First, note that the optimal solution in $\bigcup \mathcal{I}_i$ is $O(n^5/\varepsilon)$, since the maximal distance in any of $\mathcal{I}_1, \ldots, \mathcal{I}_m$ is $n^4/\varepsilon$. Using this fact, we build a simple dynamic program to prove the lemma. For all $i \leq m$ and $j \leq \log_{1+\varepsilon}/\log n(n^5/\varepsilon)$, let $k_{i,j}$ be the minimal $k'$ such that the cost of $P$ with $k'$ centers in $\mathcal{I}_i$ is at most $(1+\varepsilon/\log n)^j$. $k_{i,j}$ can be computed with a simple binary search, using the fact that the cost of a solution is decreasing with $k'$.

Given all the $k_{i,j}$, a simple dynamic program can compute $k_{\geq 1,j}$, the minimal number of centers needed to have a cost at most $(1+\varepsilon)^j$ on $\mathcal{I}_1, \ldots, \mathcal{I}_m$ (the $\varepsilon/\log n$ becomes a simple $\varepsilon$ because of the accumulation of errors). The solution for our problem is $(1+\varepsilon)^j$, where $j$ is the minimal index such that $k_{\geq 1,j} \leq k$.

The complexity of computing $k_{i,j}$ is $O(\log k \cdot n_i(\log n)^{\alpha t}(O(n^4/\varepsilon)))$, hence the complexity of computing all the $k_{i,j}$ is $O(n(\log n)^{\alpha+2}t(O(n^4/\varepsilon)))$. The complexity of the dynamic program computing $k_{\geq 1,j}$ is then simply $O(m \log n) = O(n \log n)$, which concludes the proof.

Hence, in the following, we only focus on solving problems on instances where the aspect-ratio is polynomial in $n$. 

559
Travelling on Graphs with Small Highway Dimension

Yann Disser\textsuperscript{1}, Andreas Emil Feldmann\textsuperscript{2}, Max Klimm\textsuperscript{3}, and Jochen K"onemann\textsuperscript{4}

\textsuperscript{1} TU Darmstadt, Darmstadt, Germany
disser@mathematik.tu-darmstadt.de
\textsuperscript{2} Charles University in Prague, Prague, Czechia
feldmann.a.e@gmail.com
\textsuperscript{3} Humboldt-Universität zu Berlin, Berlin, Germany
max.klimm@hu-berlin.de
\textsuperscript{4} University of Waterloo, Waterloo, Canada
jochen@uwaterloo.ca

Abstract. We study the Travelling Salesperson (TSP) and the Steiner Tree problem (STP) in graphs of low highway dimension. This graph parameter was introduced by Abraham et al. [SODA 2010] as a model for transportation networks, on which TSP and STP naturally occur for various applications in logistics. It was previously shown [Feldmann et al. ICALP 2015] that these problems admit a quasi-polynomial time approximation scheme (QPTAS) on graphs of constant highway dimension. We demonstrate that a significant improvement is possible in the special case when the highway dimension is 1, for which we present a fully-polynomial time approximation scheme (FPTAS). We also prove that STP is weakly NP-hard for these restricted graphs. For TSP we show NP-hardness for graphs of highway dimension 6, which answers an open problem posed in [Feldmann et al. ICALP 2015].

Keywords: Travelling Salesperson · Steiner Tree · Highway dimension · Approximation scheme · NP-hardness

1 Introduction

Two fundamental optimization problems already included in Karp’s initial list of 21 NP-complete problems [33] are the TRAVELLING SALESPERSON

Y. Disser—Supported by the ‘Excellence Initiative’ of the German Federal and State Governments and the Graduate School CE at TU Darmstadt.
A. E. Feldmann—Supported by the Czech Science Foundation GAČR (grant \#17-10090Y), and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004).
M. Klimm—Supported by the German Research Foundation (DFG) as part of Math\textsuperscript{+} (project AA3-4).
J. K"onemann—Supported by the Discovery Grant Program of the Natural Sciences and Engineering Research Council of Canada.

© Springer Nature Switzerland AG 2019
https://doi.org/10.1007/978-3-030-30786-8_14
problem (TSP) and the Steiner Tree problem (STP). Given an undirected graph $G = (V,E)$ with non-negative edge weights $w : E \to \mathbb{R}^+$, the TSP asks to find the shortest closed walk in $G$ visiting all nodes of $V$. Besides its fundamental role in computational complexity and combinatorial optimization, this problem has a variety of applications ranging from circuit manufacturing [29, 41] and scientific imaging [14] to vehicle routing problems [40] in transportation networks. For the STP, a subset $R \subseteq V$ of nodes is marked as terminals. The task is to find a weight-minimal connected subgraph of $G$ containing the terminals. It has plenty of fundamental applications in network design including telecommunication networks [42], computer vision [20], circuit design [30], and computational biology [22, 43], but also lies at the heart of line planning in public transportation [17].

Both TSP and STP are APX-hard in general [6, 13, 21, 34, 39, 45] implying that, unless $P = NP$, none of these problems admit a polynomial-time approximation scheme (PTAS), i.e., an algorithm that computes a $(1+\varepsilon)$-approximation in polynomial time for any given constant $\varepsilon > 0$. On the other hand, for restricted inputs PTASs do exist, e.g., for planar graphs [5, 18, 28, 36], Euclidean and Manhattan metrics [7, 44], and more generally low doubling metrics [8].

We study another class of graphs captured by the notion of highway dimension, which was proposed by Abraham et al. [3]. This graph parameter models transportation networks and is thus of particular importance in terms of applications for both TSP and STP. On a high level, the highway dimension is based on the empirical observation of Bast et al. [9, 10] that travelling from a point in a network to a sufficiently distant point on a shortest path always passes through a sparse set of “hubs”. The following formal definition is taken from [25] and follows the lines of Abraham et al. [3]. \footnote{A metric is said to have doubling dimension $d$ if for all $r > 0$ every ball of radius $r$ can be covered by at most $2^d$ balls of half the radius $r/2$.}

Here the distance between two vertices is the length of the shortest path between them, according to the edge weights. The ball $B_v(r)$ of radius $r$ around a vertex $v$ contains all vertices with distance at most $r$ from $v$.

**Definition 1.** For a scale $r \in \mathbb{R}_{>0}$, let $\mathcal{P}_{(r,2r]}$ denote the set of all vertex sets of shortest paths with length in $(r,2r]$. A shortest path cover for scale $r$ is a hitting set for $\mathcal{P}_{(r,2r]}$, i.e., a set $\text{spc}(r) \subseteq V$ such that $|\text{spc}(r) \cap P| \neq \emptyset$ for all $P \in \mathcal{P}_{(r,2r]}$. The vertices of $\text{spc}(r)$ are the hubs for scale $r$. A shortest path cover $\text{spc}(r)$ is locally $h$-sparse, if $|\text{spc}(r) \cap B_v(2r)| \leq h$ for all vertices $v \in V$. The highway dimension of $G$ is the smallest integer $h$ such that there is a locally $h$-sparse shortest path cover $\text{spc}(r)$ for every scale $r \in \mathbb{R}_{>0}$ in $G$.

The algorithmic consequences of this graph parameter were originally studied in the context of road networks [1–3], which are conjectured to have fairly small...
highway dimension. Road networks are generally non-planar due to overpasses and tunnels, and are also not Euclidean due to different driving or transmission speeds. This is even more pronounced in public transportation networks, where large stations have many incoming connections and plenty of crossing links, making Euclidean (or more generally low doubling) and planar metrics unsuitable as models. Here the highway dimension is better suited, since longer connections are serviced by larger and sparser stations (such as train stations and airports) that can act as hubs.

The main question posed in this paper is whether the structure of graphs with low highway dimension admits PTASs for problems such as TSP and STP, similar to Euclidean or planar instances. It was shown that quasi-polynomial time approximation schemes (QPTASs) exist for these problems [24], i.e., $(1 + \varepsilon)$-approximation algorithms with runtime $2^{\text{polylog}(n)}$ assuming that $\varepsilon$ and the highway dimension of the input graph are constants. However it was left open whether this can be improved to polynomial time.

1.1 Our Results

Our main result concerns graphs of the smallest possible highway dimension, and shows that for these fully polynomial time approximation schemes (FPTASs) exist, i.e., a $(1 + \varepsilon)$-approximation can be computed in time polynomial in both the input size and $1/\varepsilon$. Thus at least for this restricted case we obtain a significant improvement over the previously known QPTAS [24].

**Theorem 2.** Both Travelling Salesperson and Steiner Tree admit an FPTAS on graphs with highway dimension 1.

From an application point of view, so-called hub-and-spoke networks that can typically be seen in air traffic networks can be argued to have very small highway dimension close to 1: their star-like structure implies that hubs are needed at the centers of stars only, where all shortest paths converge. From a more theoretical viewpoint, we show that surprisingly the STP problem is non-trivial on graphs highway dimension 1, since it is still NP-hard even on this very restricted case. Interestingly, together with Theorem 2 this implies [49] that STP is weakly NP-hard on graphs of highway dimension 1. This is in contrast to planar graphs or Euclidean metrics, for which the problem is strongly NP-hard.

**Theorem 3.** The Steiner Tree problem is weakly NP-hard on graphs with highway dimension 1.\(^3\)

It was in fact left as an open problem in [24] to determine the hardness of STP and also TSP on graphs of constant highway dimension. Theorem 3 settles this question for STP. We also answer the question for TSP, but in this case we are not able to bring down the highway dimension to 1 so that the following theorem does not complement Theorem 2 tightly.

**Theorem 4.** The Travelling Salesperson problem is NP-hard on graphs with highway dimension 6.

\(^3\) The proofs of Theorems 3 and 4 are deferred to the full version of the paper.
1.2 Techniques

We present a step towards a better understanding of low highway dimension graphs by giving new structural insights on graphs of highway dimension 1. It is not hard to find examples of (weighted) complete graphs with highway dimension 1 (cf. [24]), and thus such graphs are not minor-closed. Nevertheless, it was suggested in [24] that the treewidth of low highway dimension graphs might be bounded polylogarithmically in terms of the aspect ratio $\alpha$, which is the maximum distance divided by the minimum distance between any two vertices of the input graph.

Definition 5. A tree decomposition of a graph $G = (V, E)$ is a tree $D$ where each node $v$ is labelled with a bag $X_v \subseteq V$ of vertices of $G$, such that the following holds: (a) $\bigcup_{v \in V(D)} X_v = V$, (b) for every edge $\{u, w\} \in E$ there is a node $v \in V(D)$ such that $X_v$ contains both $u$ and $w$, and (c) for every $v \in V$ the set $\{u \in V(D) \mid v \in X_u\}$ induces a connected subtree of $D$. The width of the tree decomposition is $\max\{|X_v| - 1 \mid v \in V(D)\}$. The treewidth of a graph $G$ is the minimum width among all tree decompositions for $G$.

As suggested in [24], one may hope to prove that the treewidth of any graph of highway dimension $h$ is, say, $O(h \polylog(\alpha))$. As argued in Sect. 4, it unfortunately is unlikely that such a bound is generally possible. In contrast to this, our main structural insight on graphs of highway dimension 1 is that they have treewidth $O(\log \alpha)$. This implies FPTASs for TSP and STP, since we may reduce the aspect ratio of any graph with $n$ vertices to $O(n/\varepsilon)$ and then use algorithms by Bodlaender et al. [16] to compute optimum solutions to TSP and STP in graphs of treewidth $t$ in $2^{O(t)}n$ time. Since reducing the aspect ratio distorts the solution by a factor of $1 + \varepsilon$, this results in an approximation scheme. Although these are fairly standard techniques for metrics (cf. [24]), in our case we need to take special care, since we need to bound the treewidth of the graphs resulting from this reduction, which the standard techniques do not guarantee.

It remains an intriguing open problem to understand the complexity and structure of graphs of constant highway dimension larger than 1.

1.3 Related Work

The Travelling Salesperson problem (TSP) is among Karp’s initial list of 21 NP-complete problems [33]. For general metric instances, the best known approximation algorithm is due to Christofides [23] and computes a solution with cost at most 3/2 times the LP value. For unweighted instances, the best known approximation guarantee is 7/5 and is due to Sebő and Vygen [47]. In general the problem is APX-hard [34,39,45]. For geometric instances where the nodes are points in $\mathbb{R}^d$ and distances are given by some $l_p$-norm, there exists a PTAS [4,44] for fixed $d$. When $d = \log n$, the problem is APX-hard [48]. Krauthgamer and Lee [38] generalized the PTAS to hyperbolic space. Grigni et al. [28] gave a PTAS for unweighted planar graphs which was later generalized by Arora et al. [5] to the weighted case. For improvements of the running time see Klein [36].
The Steiner Tree problem (STP) is contained in Karp’s list of NP-complete problems as well [33]. The best approximation algorithm for general metric instances is due to Byrka et al. [19] and computes a solution with cost at most $\ln(4) + \epsilon < 1.39$ times that of an LP relaxation. Their algorithm improved upon previous results by, e.g., Robins and Zelikovsky [46] and Hougardy and Prömel [32]. Also the STP is APX-hard [21] in general. For Euclidean distances and nodes in $\mathbb{R}^d$ with $d$ constant there is a PTAS due to Arora [4]. For $d = \log |R|/\log \log |R|$ where $R$ is the terminal set, the problem is APX-hard [48]. For planar graphs, there is a PTAS for STP [18], and even for the more general Steiner Forest problem for graphs with bounded genus [11]. Note that STP remains NP-complete for planar graphs [27].

It is worth mentioning that alternate definitions of the highway dimension exist. In particular, in a follow-up paper to [3], Abraham et al. [1] define a version of the highway dimension, which implies that the graphs also have bounded doubling dimension. A related model for transportation networks was given by Kosowski and Viennot [37] via the so-called skeleton dimension, which also implies bounded doubling dimension. Hence for these definitions, Bartal et al. [8] already provide a PTAS for TSP. The highway dimension definition used here (cf. Definition 1) on the other hand allows for metrics of large doubling dimension as noted by Abraham et al. [3]: a star has highway dimension 1 (by using the center vertex to hit all paths), but its doubling dimension is unbounded. While it may be reasonable to assume that road networks (which are the main concern in the works of Abraham et al. [1–3]) have low doubling dimension, there are metrics modelling transportation networks for which it can be argued that the doubling dimension is large, while the highway dimension should be small. These settings are better captured by Definition 1. For instance, the so-called hub-and-spoke networks that can typically be seen in air traffic networks are star-like networks and are unlikely to have small doubling dimension while still having very small highway dimension close to 1. Thus in these examples it is reasonable to assume that the doubling dimension is a lot larger than the highway dimension.

Feldmann et al. [24] showed that graphs with low highway dimension can be embedded into graphs with low treewidth. This embedding gives rise to a QPTAS for both TSP and STP but also other problems. However, the result in [24] is only valid for a less general definition of the highway dimension from [2], i.e., there are graphs which have constant highway dimension according to Definition 1 but for which the algorithm of [24] cannot be applied. For the less general definition from [2], Becker et al. [12] give a PTAS for BOUNDED-CAPACITY VEHICLE ROUTING in graphs of bounded highway dimension. Also the $k$-CENTER problem has been studied on graphs of bounded highway dimension, both for the less general definition [12] and the more general one used here [25, 26].

4 See [24, Section 9] and [15] for detailed discussions on different definitions of the highway dimension.
2 Structure of Graphs with Highway Dimension 1

In this section, we analyse the structure of graphs with highway dimension 1. To this end, let us fix a graph $G$ with highway dimension 1 and a shortest path cover $\text{spc}(r)$ for each scale $r \in \mathbb{R}^+$. As a preprocessing, we remove edges that are longer than the shortest path between their endpoints, so that the triangle inequality holds.

We begin by analysing the structure of the graph $G_{\leq 2r}$, which is spanned by all edges of the input graph $G$ of length at most $2r$. If $G$ has highway dimension 1 it exhibits the following key property.

**Lemma 6.** Let $G$ be a metric graph with highway dimension 1, $r \in \mathbb{R}^+$ a scale, and $\text{spc}(r)$ a shortest path cover for scale $r$. Then, every connected component of $G_{\leq 2r}$ contains at most one hub.

**Proof.** For the sake of contradiction, let $r \in \mathbb{R}^+$ and let $x, y \in \text{spc}(r)$ be a closest pair of distinct hubs in some component of $G_{\leq 2r}$. Let further $P$ be a shortest path in $G_{\leq 2r}$ between $x$ and $y$ using only edges of length at most $2r$. (Note that $P$ need not be a shortest path between $x$ and $y$ in $G$.) In particular, there is no other hub from $\text{spc}(r) \setminus \{x, y\}$ along $P$. This implies that every edge of $P$ that is not incident to either $x$ or $y$ must be of length at most $r$, since otherwise the edge would be a shortest path of length $(r, 2r]$ between its endpoints (using that $G$ is metric) contradicting the fact that $\text{spc}(r)$ is a shortest path cover for scale $r$.

Since the highway dimension of $G$ is 1, any ball $B_w(2r)$ around a vertex $w \in V(P)$ contains at most one of the hubs $x, y \in \text{spc}(r)$. Let $x', y' \in P$ be the vertices incident to $x$ and $y$ along $P$, respectively. Since the length of the edge $\{x, x'\}$ is at most $2r$, the ball $B_{x'}(2r)$ must contain $x$ and, by the observation above, it cannot contain $y$ (in particular $\{x, y\}$ is not an edge). Symmetrically, the ball $B_{y'}(2r)$ contains $y$ but not $x$. Consequently, $x' \neq y'$ and neither of these two vertices can be a hub of scale $r$, i.e., the path $P$ contains at least two vertices different from $x$ and $y$.

Let $V_x = \{w \in V : \text{dist}(x, w) < \text{dist}(y, w)\}$ contain all vertices closer to $x$ than to $y$, where $\text{dist}(\cdot, \cdot)$ refers to the distance in the original graph $G$. As all edge weights are strictly positive, we have that $\text{dist}(x, y) > 0$ and thus $y \notin V_x$. Since $P$ starts with vertex $x \in V_x$ and ends with vertex $y \notin V_x$, we deduce that there is an edge $\{u, v\}$ of $P$ such that $u \in V_x$ and $v \notin V_x$. In particular, $\text{dist}(x, u) < \text{dist}(y, u)$ and $\text{dist}(y, v) \leq \text{dist}(x, v)$. We must have $\{u, v\} \neq \{y', y\}$, since otherwise $\text{dist}(x, y') < \text{dist}(y, y') \leq 2r$ and hence $B_{y'}(2r)$ would contain $x$. Similarly, we have $\{u, v\} \neq \{x, x'\}$, since otherwise $B_{x'}(2r)$ would contain $y$. Note that, by definition, $u \neq y$ and $v \neq x$, and hence $x, y \notin \{u, v\}$. Consequently, since every edge of $P$ not incident to either $x$ or $y$ must have length at most $r$, we conclude that $\{u, v\}$ has length at most $r$.

Finally, consider the scale $r' \in \mathbb{R}^+$, defined such that $2r' = \text{dist}(x, u) + \text{dist}(u, v)$. Let $Q$ and $Q'$ denote shortest paths between $x, u$ and $v, y$ in $G$, respectively. Then the ball $B_v(2r')$ around $v$ contains $Q$ by definition of $r'$. From $\text{dist}(y, v) \leq \text{dist}(x, v) \leq \text{dist}(x, u) + \text{dist}(u, v) = 2r'$ it follows that
$B_v(2r')$ contains $Q'$ as well. Also, $\dist(y,v) \leq \dist(x,v)$ means that $B_v(2r)$ cannot contain $x$, and hence $2r' = \dist(x,u) + \dist(u,v) \geq \dist(x,v) > 2r$, which implies $r' > r$. W.l.o.g., assume that $\dist(x,u) \leq \dist(v,y)$ (otherwise consider scale $2r' = \dist(y,v) + \dist(u,v)$ and the ball $B_u(2r')$). Our earlier observation that $\dist(u,v) \leq r$ with $r < r'$ then yields $\dist(v,y) \geq \dist(x,u) = 2r' - \dist(u,v) > r'$. In other words, the lengths of both paths $Q$ and $Q'$ are in $(r',2r']$, and so they both need to contain a hub of $\text{spc}(r')$. However, by definition of $u,v$, the paths $Q$ and $Q'$ are vertex disjoint, which means that the ball $B_v(2r')$, which contains $Q$ and $Q'$, also contains at least two hubs from $\text{spc}(r')$. This is a contradiction with $G$ having highway dimension 1. \hfill \Box

Given a graph $G$, we now consider graphs $G_{\leq 2r}$ for exponentially growing scales. In particular, for any integer $i \geq 0$ we define the scale $r_i = 2^i$ and call a connected component of $G_{< 2r_i}$ a level-$i$ component. Note that the level-$i$ components partition the graph $G$, and that the level-$i$ components are a refinement of the level-$(i + 1)$ components, i.e., every level-$i$ component is contained in some level-$(i + 1)$ component. W.l.o.g., we scale the edge weights of the graph such that $\min_{e \in E} w(e) = 3$, so that there are no edges on level 0, and every level-0 component is a singleton. Let $\alpha = \max_{u \neq v} \frac{\dist(u,v)}{\min_{u \neq v} \dist(u,v)} = \max_{u \neq v} \frac{\dist(u,v)}{3}$ be the aspect ratio of $G$. In our applications we may assume that $G$ is connected, so that there is exactly one level-$\left(1 + \lceil \log_2(\alpha) \rceil\right)$ component containing all of $G$.

Since every edge is a shortest path between its endpoints, every edge $e = \{u,v\}$ that connects a vertex $u$ of a level-$i$ component $C$ with a vertex $v$ outside $C$ is hit by a hub of $\text{spc}(r_j)$, where $j$ is the level for which $w(e) \in (r_j,2r_j]$. Moreover, since $v$ lies outside $C$, we have $w(e) > 2r_i$ and, thus, $j \geq i + 1$. The following definition captures the set of the hubs through which edges can possibly leave $C$.

**Definition 7.** Let $C$ be a level-$i$ component of $G$. We define the set of interface points of $C$ as $I_C := \bigcup_{j \geq i} \{u \in \text{spc}(r_j) : \text{dist}_C(u) \leq 2r_j\}$, where $\text{dist}_C(u)$ denotes the minimum distance from $u$ to a vertex in $C$ (if $u \in C$, $\text{dist}_C(u) = 0$).

Note that, for technical reasons, we explicitly add every hub at level $i$ of a component to its set of interface points as well, even if such a hub does not connect the component with any vertex outside at distance more than $2r_i$.

**Lemma 8.** If $G$ has highway dimension 1, then each interface $I_C$ of a level-$i$ component $C$ contains at most one hub for each level $j \geq i$.

**Proof.** Assume that there are two hubs $u,v \in \text{spc}(r_j)$ in $I_C$, and recall that we preprocessed the graph so that the triangle inequality holds. Then $u$ and $v$ must be contained in the same level-$j$ component $C'$, since $u$ and $v$ are connected to $C$ with edges of length at most $2r_j$ (or are contained in $C$) and $C \subseteq C'$. This contradicts Lemma 6. \hfill \Box

Using level-$i$ components and their interface points we can prove that the treewidth of a graph with highway dimension 1 is bounded in terms of the aspect ratio.
Lemma 9. If a graph $G$ has highway dimension 1 and aspect ratio $\alpha$, its treewidth is at most $1 + \lceil \log_2(\alpha) \rceil$.

Proof. The tree decomposition of $G$ is given by the refinement property of level-$i$ components. That is, let $D$ be a tree that contains a node $v_C$ for every level-$i$ component $C$ for all levels $0 \leq i \leq 1 + \lceil \log_2(\alpha) \rceil$. For every node $v_C$ we add an edge in $D$ to node $v_{C'}$, if $C$ is a level-$i$ component, $C'$ is a level-$(i+1)$ component, and $C \subseteq C'$. The bag $X_C$ for node $v_C$ contains the interface points $I_C$. For a level-0 component $C$ the bag $X_C$ additionally contains the single vertex $u$ contained in $C$.

Clearly, the tree decomposition has Property (a) of Definition 5, since the level-0 components partition the vertices of $G$ and every vertex of $G$ is contained in a bag $X_C$ corresponding to a level-0 component $C$. Also, Property (b) is given by the bags $X_C$ for level-0 components $C$, since for every edge $e$ of $G$ one of its endpoints $u$ is a hub of $\text{spc}(r_i)$ where $i$ is such that $w(e) \in (r_i, 2r_i]$, and the other endpoint $w$ is contained in a level-0 component $C$, for which $X_C$ contains $u$ and $w$.

For Property (c), first consider a vertex $u$ of $G$, which is not contained in any set of interface points for any level-$i$ component and any $0 \leq i \leq \log_2(\alpha)$. Such a vertex only appears in the bag $X_C$ for the level-0 component $C$ containing $u$, and thus the node $v_C$ for which the bag contains $u$ trivially induces a connected subtree of $D$.

Any other vertex $u$ of $G$ is an interface point. Let $i$ be the highest level for which $u \in I_C$ for some level-$i$ component $C$. We claim that $u \in C$, which implies that $C$ is the unique level-$i$ component containing $u$ in its interface. To show our claim, assume $u \notin C$. Then, by definition, $I_C$ contains $u$ because $u \in \text{spc}(r_j)$ for some $j \geq i$ and $u$ has some neighbour at distance at most $2r_j$ in $C$. Since we preprocessed the graph such that every edge is a shortest path between its endpoints, this means that there must be an edge $e = \{u, v\}$ with $w(e) \in (r_j, 2r_j]$ and $v \in C$. Since $u \notin C$, we have $i < j$. Let $C'$ be the unique level-$j$ component with $C \subseteq C'$. Then, by definition, $u \in I_{C'}$, which contradicts the maximality of $i$. This proves our claim and shows that the highest level component $C$ with $u \in X_C$ is uniquely defined. Moreover, we obtain $u \in \text{spc}(r_i)$.

Now consider a level-$i'$ component $C'$ with $i' < i$, such that $u \in X_{C'}$, and let $C''$ be the unique level-$(i' + 1)$ component containing $C'$. We claim that $u \in X_{C''}$. If $u \in C' \subseteq C''$, then $u \in X_{C''}$, since $u \in \text{spc}(r_i)$, $\text{dist}_{C''}(u) = 0 \leq 2r_i$ and $i' + 1 \leq i$. If $u \notin C'$, then $u \in X_{C'}$ implies $u \in I_{C'}$, which means that there must be a vertex $w \in C'$ with $\text{dist}(u, w) \leq 2r_i$. But then $w \in C''$ and thus $\text{dist}_{C''}(u) \leq 2r_i$. Together with $u \in \text{spc}(r_i)$, this implies $u \in X_{C''}$, as claimed. Since $v_{C''}$ is a child of $v_{C'}$ in the tree $D$, it follows inductively that the nodes of $D$ with bags containing $u$ induce a subtree of $D$ with root $v_C$, which establishes Property (c).

By Lemma 8 each set of interface points contains at most one hub of each level. Since all edges have length at least 3, there are no hubs in $\text{spc}(r_0)$ on level 0. This means that each bag of the tree decomposition contains at most $1 + \lceil \log_2(\alpha) \rceil$
An additional property that we will exploit for our algorithms is the following. A \((\mu, \delta)-\text{net}\) \(N \subseteq V\) is a subset of vertices such that (a) the distance between any two distinct net points \(u, w \in N\) is more than \(\mu\), and (b) for every vertex \(v \in V\) there is some net point \(w \in N\) at distance at most \(\delta\). For graphs of highway dimension 1 however, we can obtain nets with additional favourable properties, as the next lemma shows.

**Lemma 10.** For any graph \(G\) of highway dimension 1 and any \(r > 0\), there is an \((r, 3r)\)-net such that every connected component of \(G_{\leq r}\) contains exactly one net point. Moreover this net can be computed in polynomial time.

**Proof.** We first derive an upper bound of \(3r\) for the diameter of any connected component of \(G_{\leq r}\). Lemma 6 implies that a connected component \(C\) contains at most one hub \(x\) of \(\text{spc}(r/2)\). By definition, any shortest path in \(C\) of length in \((r/2, r]\) must pass through \(x\). We also know that every edge of \(C\) has length at most \(r\). Consequently, every edge in \(C\) not incident to \(x\) must have length at most \(r/2\), since each edge constitutes a shortest path between its endpoints. This implies that any shortest path in \(C\) that is not hit by \(x\) must have length at most \(r/2\): if \(C\) contains a shortest path \(P\) with length more than \(r/2\) not containing \(x\) we could repeatedly remove edges of length at most \(r/2\) from \(P\) until we obtain a shortest path of length in \((r/2, r]\) not hit by \(x\), a contradiction.

Now consider a shortest path \(P\) in \(G\) of length more than \(r/2\) from some vertex \(v \in C\) to \(x\) (note that this path may not be entirely contained in \(C\)). Let \(\{u, w\}\) be the unique edge of \(P\) such that \(\text{dist}(v, u) \leq r/2\) and \(\text{dist}(v, w) > r/2\). If the length of the edge \(\{u, w\}\) is at most \(r/2\) then \(\text{dist}(v, w) \leq r\), and thus \(w = x\), since the part of the path from \(v\) to \(w\) is a shortest path of length in \((r/2, r]\) and thus needs to pass through \(x\). Otherwise the length of the edge \(\{u, w\}\) is in the interval \((r/2, r]\), which again implies \(w = x\), since the edge must contain \(x\). In either case, \(\text{dist}(v, x) \leq 3r/2\). This implies that every vertex in \(C\) is at distance at most \(3r/2\) from \(x\), and thus the diameter of \(C\) is at most \(3r\).

To compute the \((r, 3r)\)-net, we greedily pick an arbitrary vertex of each connected component of \(G_{\leq r}\). As the distances between components of \(G_{\leq r}\) is greater than \(r\), and every vertex lies in some component containing a net point, we get the desired distance bounds. Clearly this net can be computed in polynomial time.

\[\square\]

## 3 Approximation Schemes

In general the aspect ratio of a graph may be exponential in the input size. A key ingredient of our algorithms is to reduce the aspect ratio \(\alpha\) of the input graph \(G = (V, E)\) to a polynomial. For STP and TSP, standard techniques can be used to reduce the aspect ratio to \(O(n/\varepsilon)\) when aiming for a \((1+\varepsilon)\)-approximation. This was for instance also used in [24] for low highway dimension graphs, but here
we need to take special care not to destroy the structural properties given by Lemma 9 in this process. In particular, we need to reduce the aspect ratio and maintain the fact that the treewidth is bounded.

Therefore, we reduce the aspect ratio of our graphs by the following preprocessing. Both metric TSP and STP admit constant factor approximations in polynomial time using well-known algorithms [19,23]. We first compute a solution of cost $c$ using a $\beta$-approximation algorithm for the problem at hand (TSP or STP). For TSP, the diameter of the graph $G$ clearly is at most $c/2$. For STP we remove every vertex of $V$ that is at distance more than $c$ from any terminal, since such a vertex cannot be part of the optimum solution. After having removed all such vertices in this way, we obtain a graph $G$ of diameter at most $3c$. Thus, in the following, we may assume that our graph $G$ has diameter at most $3c$. We then set $r = \frac{\epsilon c}{3n}$ in Lemma 10 to obtain a $(\frac{\epsilon c}{n}, \frac{\epsilon c}{n})$-net $N \subseteq V$. As a consequence the metric induced by $N$ (with distances of $G$) has aspect ratio at most $\frac{3c}{\epsilon c/(3n)} = O(n/\epsilon)$, since the minimum distance between any two net points of $N$ is at least $\frac{\epsilon c}{3n}$ and the maximum distance is at most $3c$. We will exploit this property in the following.

By Lemma 10, each connected component of $G < \frac{\epsilon c}{n}$ contains exactly one net point of $N$. Let $\eta: V \mapsto N$ map each vertex of $G$ to the unique net point in the same connected component of $G < \frac{\epsilon c}{n}$. We define a new graph $G'$ with vertex set $N \subseteq V$ and edge set $\{\{\eta(u), \eta(v)\} : \{u, v\} \in E \land \eta(u) \neq \eta(v)\}$. The length of each edge $\{w, w'\}$ of $G'$ is the shortest path distance between $w$ and $w'$ in $G$. This new graph $G'$ may not have bounded highway dimension, but we claim that it has treewidth $O(\log(n/\epsilon))$.

**Lemma 11.** If $G$ has highway dimension 1, the graph $G'$ with vertex set $N$ has treewidth $O(\log(n/\epsilon))$. Moreover, a tree decomposition for $G'$ of width $O(\log(n/\epsilon))$ can be computed in polynomial time.

**Proof.** We construct a tree decomposition $D'$ of $G'$ as follows. Following Lemma 9 we can compute a tree decomposition $D$ of width at most $1 + \lceil \log_2(\alpha) \rceil$, where $\alpha$ is the aspect ratio of $G$: for this we need to compute a locally 1-sparse shortest path cover $\text{spc}(r_i)$ for each level $i$, which can be done in polynomial time via an XP algorithm [24] if the highway dimension is 1. We then find the level-$i$ components and their interface points, from which the tree decomposition $D$ and its bags can be constructed. Since there are $O(\log \alpha)$ levels and $\alpha$ is at most exponential in the input size (which includes the encoding length of the edge weights), we can compute $D$ in polynomial time.

We construct $D'$ from $D$ by replacing every bag $X$ of $D$ by a new bag $X' = \{\eta(v) : v \in X\}$ containing the net points for the vertices in $X$. It is not hard to see that Properties (a) and (b) of Definition 5 are fulfilled by $D'$, since they are true for $D$. For Property (c), note that for any edge $\{u, v\}$ of $G$, the set of all bags of $D$ that contain $u$ or $v$ form a connected subtree of $D$. This is because the bags containing $u$ form a connected subtree (Property (c)), the same is true for $v$, and both these subtrees share at least one node labelled by a bag containing the edge $\{u, v\}$ (Property (b)). Consequently, the set of
all bags containing vertices of any connected subgraph of $G$ form a connected subtree. In particular, for any connected component $A$ of $G \leq \frac{\varepsilon c}{3n}$, the set of bags of $D$ containing at least one vertex of $A$ form a connected subtree. This implies Property (c) for $D'$. Thus, $D'$ is indeed a tree decomposition of $G'$ according to Definition 5. Note that $D'$ can be computed in polynomial time.

To bound the width of $D'$, recall that a bag $X$ of the tree decomposition $D$ of $G$ contains the interface points $I_C$ of a level-$i$ component $C$, in addition to one more vertex of $C$ on the lowest level $i = 0$. Each interface point is a hub from $\text{SPC}(r_j)$ at some level $j \geq i$ and is at distance at most $2r_j$ from $C$. In particular, if $2r_i \leq \frac{\varepsilon c}{3n}$ then $C$ is a component of $G \leq 2r_i \subseteq G \leq \frac{\varepsilon c}{3n}$, and all hubs of $I_C \cap \text{SPC}(r_j)$ for which $2r_j \leq \frac{\varepsilon c}{3n}$ lie in the same connected component $A$ of $G \leq \frac{\varepsilon c}{3n}$ as $C$. These hubs are therefore all mapped to the same net point $w$ in $A$ by $\eta$. In addition to $w$, the bag $X' = \{ \eta(v) : v \in X \}$ resulting from $X$ and $\eta$ contains at most one vertex for every level $j$ such that $2r_j > \frac{\varepsilon c}{3n}$. As $r_j = 2^j$, this condition is equivalent to $j > \log_2(\frac{\varepsilon c}{3n}) - 1$. As there are $1 + \lceil \log_2(\alpha) \rceil$ levels in total, there are $O(\log(\frac{\alpha n}{\varepsilon c}))$ hubs in $X'$. This bound is obviously also valid in case $2r_i > \frac{\varepsilon c}{3n}$.

We preprocessed the graph $G$ so that its diameter is at most $3c$ and its minimum distance is 3, which implies an aspect ratio $\alpha$ of at most $c$ for $G$. This means that every bag $X'$ contains $O(\log(n/\varepsilon))$ vertices, and thus the claimed treewidth bound for $G'$ follows.

We are now ready to prove our main result.

Proof (of Theorem 2). To solve TSP or STP on $G$ we first use the above reduction to obtain $G'$ and its tree decomposition $D'$, and then compute an optimum solution for $G'$. For TSP, $G'$ is already a valid input instance, but for STP we need to define a terminal set, which simply is $R' = \{ \eta(v) : v \in R \}$ if $R$ is the terminal set of $G$. Bodlaender et al. [16] proved that for both TSP and STP there are deterministic algorithms to solve these problems exactly in time $2^{O(t)n}$, given a tree decomposition of the input graph of width $t$. By Lemma 11 we can thus compute the optimum to $G'$ in time $2^{O(\log(n/\varepsilon))} \cdot n = (n/\varepsilon)^{O(1)}$. Afterwards, we convert the solution for $G'$ back to a solution for $G$, as follows.

For TSP we may greedily add vertices of $V$ to the tour on $N$ by connecting every vertex $v \in V$ to the net point $\eta(v)$. As the vertices $N$ of $G'$ form a $(\frac{\varepsilon c}{3n}, \frac{\varepsilon c}{n})$-net of $V$, this incurs an additional cost of at most $2\frac{\varepsilon c}{n}$ per vertex, which sums up to at most $2\varepsilon c$. Let $\text{OPT}$ and $\text{OPT}'$ denote the costs of the optimum tours in $G$ and $G'$, respectively. We know that $c \leq \beta \cdot \text{OPT}$, since we used a $\beta$-approximation algorithm to compute $c$. Furthermore, the optimum tour in $G$ can be converted to a tour in $G'$ of cost at most $\text{OPT}$ by short-cutting, due to the triangle inequality. Thus $\text{OPT}' \leq \text{OPT}$, which means that the cost of the computed tour in $G$ is at most $\text{OPT}' + 2\varepsilon c \leq (1 + 2\beta \varepsilon)\text{OPT}$.

Similarly, for STP we may greedily connect a terminal $v$ of $G$ to the terminal $\eta(v)$ of $G'$ in the computed Steiner tree in $G'$. This adds an additional cost of at most $\frac{\varepsilon c}{n}$, which sums up to at most $\varepsilon c$. Let now $\text{OPT}$ and $\text{OPT}'$ be the costs of the optimum Steiner trees in $G$ and $G'$, respectively. We may convert a Steiner tree $T$ in $G$ into a tree $T'$ in $G'$ by using edge $\{ \eta(u), \eta(v) \}$ for each edge $\{ u, v \}$ of $T$. Note
that the resulting tree $T'$ contains all terminals of $G'$, since $R' = \{\eta(v) \mid v \in R\}$. As the vertices $N$ of $G'$ form a $(\frac{\varepsilon c}{3n}, \frac{\varepsilon c}{n})$-net of $V$, the cost of $T'$ is at most $\text{OPT} + 2\varepsilon c$ if the cost of $T$ is $\text{OPT}$ (by the same argument as used for the proof of Lemma 11). As before, we know that $c \leq \beta \cdot \text{OPT}$, and thus the cost of the computed Steiner tree in $G$ is at most $\text{OPT}' + \varepsilon c \leq \text{OPT} + 3\varepsilon c \leq (1 + 3\beta \varepsilon)\text{OPT}$.

Hence we obtain FPTASs for both TSP and STP, which compute $(1 + \varepsilon)$-approximations within a runtime that is polynomial in the input size and $1/\varepsilon$.

\[\Box\]

4 Conclusions

We showed that, somewhat surprisingly, graphs of highway dimension 1 exhibit a rich combinatorial structure. On one hand, it was already known [24] that these graphs are not minor-closed and thus their treewidth is unbounded. Here we additionally showed that STP is weakly NP-hard on such graphs, further confirming that these graphs have non-trivial properties. On the other hand, we proved in Lemma 9 that the treewidth of a graph of highway dimension 1 is logarithmically bounded in the aspect ratio $\alpha$. This in turn can be exploited to obtain a very efficient FPTAS for both STP and TSP.

At this point one may wonder whether it is possible to generalize Lemma 9 to larger values of the highway dimension. In particular, in [24] it was suggested that the treewidth of a graph of highway dimension $h$ might be bounded by, say, $O(h \text{polylog}(\alpha))$. However such a bound is highly unlikely in general, since it would have the following consequence for the $k$-CENTER problem, for which $k$ vertices (centers) need to be selected in a graph such that the maximum distance of any vertex to its closest center is minimized. It was shown in [25] that it is NP-hard to compute a $(2 - \varepsilon)$-approximation for $k$-CENTER on graphs of highway dimension $O(\log^2 n)$, for any $\varepsilon > 0$. Given such a graph, the same preprocessing of Sect. 3 could be used to derive an analogue of Lemma 11, i.e., a graph $G'$ of treewidth $O(\text{polylog}(n/\varepsilon))$ could be computed for the net $N$. Moreover, a 2-approximation for $k$-CENTER can be computed in polynomial time on any graph [31], and if the input has treewidth $t$ a $(1 + \varepsilon)$-approximation can be computed in $(t/\varepsilon)^{O(t)} n^{O(1)}$ time [35]. Using the same arguments to prove Theorem 2 for STP and TSP, it would now be possible to compute a $(1 + \varepsilon)$-approximation for $k$-CENTER in quasi-polynomial time (cf. [26]). That is, we would obtain a QPTAS for graphs of highway dimension $O(\log^2 n)$, which is highly unlikely given that computing a $(2 - \varepsilon)$-approximation is NP-hard on such graphs.

The above argument rules out any bound of $(h \log \alpha)^{O(1)}$ for graphs of highway dimension $h$ and aspect ratio $\alpha$, unless NP-hard problems admit quasi-polynomial time algorithms. In fact, we conjecture that the $k$-CENTER problem is NP-hard to approximate within a factor of $2 - \varepsilon$ for graphs of constant highway dimension (for some constant larger than 1). If this is true, then the above argument even rules out a treewidth bound of $f(h) \text{polylog}(\alpha)$ for any function $f$. Thus, in order to answer the open problem of [24] and obtain a PTAS for graphs of constant highway dimension, a different approach seems to be needed.
References

47. Sebő, A., Vygen, J.: Shorter tours by nicer ears: \(7/5\)-approximation for the graph-TSP, \(3/2\) for the path version, and \(4/3\) for two-edge-connected subgraphs. Combinatorica 34, 1–34 (2014)
Abstract

We study the Steiner Tree problem, in which a set of terminal vertices needs to be connected in the cheapest possible way in an edge-weighted graph. This problem has been extensively studied from the viewpoint of approximation and also parametrization. In particular, on one hand Steiner Tree is known to be APX-hard, and W[2]-hard on the other, if parameterized by the number of non-terminals (Steiner vertices) in the optimum solution. In contrast to this we give an efficient parameterized approximation scheme (EPAS), which circumvents both hardness results. Moreover, our methods imply the existence of a polynomial size approximate kernelization scheme (PSAKS) for the considered parameter.

We further study the parameterized approximability of other variants of Steiner Tree, such as Directed Steiner Tree and Steiner Forest. For neither of these an EPAS is likely to exist for the studied parameter: for Steiner Forest an easy observation shows that the problem is APX-hard, even if the input graph contains no Steiner vertices. For Directed Steiner Tree we prove that computing a constant approximation for this parameter is W[1]-hard. Nevertheless, we show that an EPAS exists for Unweighted Directed Steiner Tree. Also we prove that there is an EPAS and a PSAKS for Steiner Forest if in addition to the number of Steiner vertices, the number of connected components of an optimal solution is considered to be a parameter.

2012 ACM Subject Classification Mathematics of computing → Combinatorics, Mathematics of computing → Graph theory, Theory of computation → Design and analysis of algorithms

Keywords and phrases Steiner Tree, Steiner Forest, Approximation Algorithms, Parameterized Algorithms, Lossy Kernelization
1 Introduction

In this paper we study several variants of the Steiner Tree problem. In its most basic form this optimization problem takes an undirected graph $G = (V, E)$ with edge weights $w(e) \in \mathbb{R}_+^+$ for every $e \in E$, and a set $R \subseteq V$ of terminals as input. The non-terminals in $V \setminus R$ are called Steiner vertices. A Steiner tree is a tree in the graph $G$, which spans all terminals in $R$ and may contain some of the Steiner vertices. The objective is to minimize the total weight $\sum_{e \in E(T)} w(e)$ of the computed Steiner tree $T \subseteq G$. This fundamental optimization problem is one of the 21 original NP-hard problems listed by Karp \cite{Karp1972} in his seminal paper from 1972, and has been intensively studied since then. The Steiner Tree problem and its variants have applications in network design, circuit layouts, and phylogenetic tree reconstruction, among others (see survey \cite{Nguyen2020}).

Two popular ways to handle the seeming intractability of NP-hard problems are to design approximation \cite{Johnson1974} and parameterized \cite{FlumGrohe2006} algorithms. For the former, an $\alpha$-approximation is computed in polynomial time for some factor $\alpha$ specific to the algorithm, i.e., the solution is always at most a multiplicative factor of $\alpha$ worse than the optimum of the input instance. The Steiner Tree problem, even in its basic form as defined above, is APX-hard \cite{Karp1972}, i.e., it is NP-hard to obtain an approximation factor of $\alpha = \frac{\ln(4)}{\ln(3)} \approx 1.01$. However a factor of $\alpha = \ln(4) + \varepsilon \approx 1.39$ can be achieved in polynomial time \cite{Bjorklund2012}, which is the currently best factor known for this runtime.

For parameterized algorithms, an instance is given together with a parameter $p$ describing some property of the input. The optimum solution is computed in time $f(p) \cdot n^{O(1)}$, where $f$ is a computable function independent of the input size $n$. If such an algorithm exists, we call the problem fixed-parameter tractable (FPT) for parameter $p$. A well-studied parameter for the Steiner Tree problem is the number of terminals $|R|$. It is known since the classical result of Dreyfus and Wagner \cite{DreyfusWagner1971} that the Steiner Tree problem is FPT for this parameter, as the problem can be solved in time $3^{|R|} \cdot n^{O(1)}$ if $n = |V|$. This can be improved to $2^{|R|} \cdot n^{O(1)}$ if the input graph is unweighted using the results of Björklund et al. \cite{Bjorklund2010}. A somewhat complementary and less-studied parameter to the number of terminals is the number of Steiner vertices in the optimum solution, i.e., $p = |V(T) \setminus R|$ if $T$ is an optimum Steiner tree. It is known \cite{Heggernes2010} that Steiner Tree is $W[2]$-hard for parameter $p$ and therefore is unlikely to be FPT, in contrast to the parameter $|R|$. This parameter $p$ has been mainly studied in the context of unweighted graphs before. The problem remains $W[2]$-hard in this special case and therefore the focus has been on designing parameterized algorithms for restricted graph classes, such as planar or $d$-degenerate graphs \cite{Dvořák2015, Heggernes2010}.
In contrast to this, our question is: what can be done in the most general case, in which the class of input graphs is unrestricted and edges may have weights? Our main result is that we can overcome the APX-hardness of Steiner Tree on one hand, and on the other hand also the \( W[2]\)-hardness for our parameter of choice \( p \), by combining the two paradigms of approximation and parametrization. This relatively new and growing area has gained quite a bit of attention recently (see e.g., [3, 6, 8, 9, 10, 18, 20, 26, 27, 28, 29, 32, 34]). We show that there is an efficient parameterized approximation scheme (EPAS), which for any \( \varepsilon > 0 \) computes a \((1 + \varepsilon)\)-approximation in time \( f(p, \varepsilon) \cdot n^{O(1)} \) for a computable function \( f \) independent of \( n \). Note that here we consider the approximation factor of the algorithm as a parameter as well, which accounts for the “efficiency” of the approximation scheme (analogous to an efficient polynomial time approximation scheme or EPTAS). In fact, as summarized in the following theorem, our algorithm computes an approximation to the cheapest tree having at most \( p \) Steiner vertices, even if better solutions with more Steiner vertices exist.

\[ \text{Theorem 1.} \quad \text{There is an algorithm for Steiner Tree, which given an edge-weighted undirected graph } G = (V, E), \text{ terminal set } R \subseteq V, \varepsilon > 0, \text{ and integer } p, \text{ computes a } (1 + \varepsilon)\text{-approximation to the cheapest Steiner tree } T \subseteq G \text{ with } p \geq |V(T) \setminus R| \text{ in time } 2^{O(p^2/\varepsilon^2)} \cdot n^{O(1)}. \]

Many variants of the Steiner Tree problem exist, and we explore the applicability of our techniques to some common ones. For the Directed Steiner Tree problem the aim is to compute an arborescence, i.e., a directed graph obtained by orienting the edges of a tree so that exactly one vertex called the root has in-degree zero (which means that all vertices are reachable from the root). More concretely, the input consists of a directed graph \( G = (V, A) \) with arc weights \( w(a) \in \mathbb{R}_+^k \) for every \( a \in A \), a terminal set \( R \subseteq V \), and a specified terminal \( r \in R \). A Steiner arborescence is an arborescence in \( G \) with root \( r \) containing all terminals \( R \). The objective is to find a Steiner arborescence \( T \subseteq G \) minimizing the weight \( \sum_{a \in A(T)} w(a) \).

This problem is notoriously hard to approximate: no \( O((\log^{2-\varepsilon}(n)))\)-approximation exists unless \( \text{NP} \subseteq \text{ZTIME}(n^{polylog(n)}) \) [22]. But even for the Unweighted Directed Steiner Tree problem in which each arc has unit weight, a fairly simple reduction from the Set Cover problem implies that no \((1 - \varepsilon \ln n)\)-approximation algorithm is possible unless \( P = \text{NP} \) [13, 22]. At the same time, even Unweighted Directed Steiner Tree is \( W[2] \)-hard for our considered parameter \( p \) [24, 30], just as for the undirected case. For this reason, all previous results have focused on restricted inputs: Jones et al. [24] prove that when combining the parameter \( p \) with the size of the largest excluded topological minor of the input graph, Unweighted Directed Steiner Tree is \( \text{FPT} \). They also show that if the input graph is acyclic and \( d \)-degenerate, the problem is \( \text{FPT} \) for the combined parameter \( p \) and \( d \).

Our focus again is on general unrestricted inputs. We are able to leverage our techniques to the unweighted directed setting, and obtain an EPAS, as summarized in the following theorem. Here the cost of a Steiner arborescence is the number of contained arcs.

\[ \text{Theorem 2.} \quad \text{There is an algorithm for Unweighted Directed Steiner Tree, which given an unweighted directed graph } G = (V, A), \text{ terminal set } R \subseteq V, \text{ root } r \in R, \varepsilon > 0, \text{ and integer } p, \text{ computes a } (1 + \varepsilon)\text{-approximation to the cheapest Steiner arborescence } T \subseteq G \text{ with } p \geq |V(T) \setminus R| \text{ in time } 2^{O(p^2/\varepsilon^2)} \cdot n^{O(1)}. \]

1 If the input to this optimization problem is malformed (e.g., if \( p \) is smaller than the number of Steiner vertices of any feasible solution) then the output of the algorithm can be arbitrary (cf. [28]).
Can our techniques be utilized for the even more general case when arcs have weights? Interestingly, in contrast to the above theorem we can show that in general the Directed Steiner Tree problem most likely does not admit such approximation schemes, even when allowing “non-efficient” runtimes of the form \( f(p, \varepsilon) \cdot g(\varepsilon) \) for any computable functions \( f \) and \( g \). This follows from the next theorem, since setting \( \varepsilon \) to any constant, the existence of such a \((1+\varepsilon)\)-approximation algorithm would imply \( W[1] = FPT \).

\[ \text{Theorem 3.} \quad \text{For any constant } \alpha, \text{ it is } W[1]-\text{hard to compute an } \alpha\text{-approximation of the optimum Steiner arborescence } T \text{ for Directed Steiner Tree parameterized by } |V(T) \setminus R|, \text{ if the input graph is arc-weighted.} \]

Other common variants of Steiner Tree include the Prize Collecting Steiner Tree and Steiner Forest problems. The latter takes as input an edge-weighted undirected graph \( G = (V, E) \) and a list \( \{s_1, t_1\}, \ldots, \{s_k, t_k\} \) of terminal pairs, i.e., \( R = \{s_i, t_i \mid 1 \leq i \leq k\} \). A Steiner forest is a forest \( F \) in \( G \) for which each \( \{s_i, t_i\} \) pair is in the same connected component, and the objective is to minimize the total weight of the forest \( F \). For this variant it is not hard to see that parametrizing by \( p = |V(F) \setminus R| \) cannot yield any approximation scheme, as a simple reduction from Steiner Tree shows that the problem is APX-hard even if the input has no Steiner vertices (cf. [17]). For the Prize Collecting Steiner Tree problem, the input is again a terminal set in an edge-weighted graph, but the terminals have additional costs. A solution tree is allowed to leave out a terminal but has to pay its cost in return (cf. [35]). It is also not hard to see that this problem is APX-hard, even if there are no Steiner vertices at all. These simple results show that our techniques to obtain approximation schemes reach their limit quite soon: with the exception of Unweighted Directed Steiner Tree, most common variants of Steiner Tree seem not to admit approximation schemes for our parameter \( p \). We are however able to generalize our EPAS to Steiner Forest if we combine \( p \) with the number \( c \) of connected components in the optimum solution. In fact, our main result of Theorem 1 is a corollary of the next theorem, using only the first part of the above mentioned reduction from Steiner Tree. Due to this, it is not possible to have a parameterized approximation scheme for the parameter \( c \) alone, as such an algorithm would imply a polynomial time approximation scheme for the APX-hard Steiner Tree problem. Hence the following result necessarily needs to combine the parameters \( p \) and \( c \).

\[ \text{Theorem 4.} \quad \text{There is an algorithm for Steiner Forest, which given an edge-weighted undirected graph } G = (V, E), \text{ a list } \{s_1, t_1\}, \ldots, \{s_k, t_k\} \subseteq V \text{ of terminal pairs}, \varepsilon > 0, \text{ and integers } p, c, \text{ computes a } (1+\varepsilon)\text{-approximation to the cheapest Steiner forest } F \subseteq G \text{ with at most } c \text{ connected components and } p \geq |V(F) \setminus R| \text{ where } R = \{s_i, t_i \mid 1 \leq i \leq k\}, \text{ in time } (2c)^O((p+c)^2/c^4) \cdot R^{O(1)}. \]

A topic tightly connected to parameterized algorithms is kernelization. We here use the framework of Lokshtanov et al. [28], who also give a thorough introduction to the topic. Loosely speaking, a kernelization algorithm runs in polynomial time, and, given an instance of a parameterized problem, computes another instance of the same problem, such that the size of the latter instance is at most \( f(p) \) for some computable function \( f \) in the parameter \( p \) of the input instance. The computed instance is called the kernel, and for an optimization problem it must be possible to efficiently convert an optimum solution to the kernel into an optimum solution to the input instance.

A fundamental result of parameterized complexity says that a problem is FPT if and only if it has a kernelization algorithm [12]. This means that for our parameter \( p \), most likely
Steiner Tree does not have a kernelization algorithm, as it is \( W[2] \)-hard. For this reason, the focus of kernelization results has previously again shifted to special cases. By a folklore result, Steiner Tree is \( \text{FPT} \) for our parameter \( p \) if the input graph is planar (cf. [24]). Of particular interest are polynomial kernels, which have size polynomial in the input parameter. The idea is that computing the kernel in this case is an efficient preprocessing procedure for the problem, such that exhaustive search algorithms can be used on the kernel. Suchý [33] proved that Unweighted Steiner Tree parameterized by \( p \) admits a polynomial kernel if the input graph is planar.

Our aspirations again are to obtain results for inputs that are as general as possible, i.e., on unrestricted edge-weighted input graphs. We prove that Steiner Tree has a polynomial lossy (approximate) kernel, despite the fact that the problem is \( W[2] \)-hard: an \( \alpha \)-approximate kernelization algorithm is a kernelization algorithm that computes a new instance for which a given \( \beta \)-approximation can be converted into an \( \alpha \beta \)-approximation for the input instance in polynomial time. The new instance is now called a (polynomial) approximate kernel, and its size is again bounded as a function (a polynomial) of the parameter of the input instance.

Just as for our parameterized approximation schemes in Theorems 1 and 4, we prove the existence of a lossy kernel for Steiner Tree by a generalization to Steiner Forest where we combine the parameter \( p \) with the number \( c \) of connected components in the optimum solution. Also, our lossy kernel can approximate the optimum arbitrarily well: we prove that for our parameter the Steiner Forest problem admits a polynomial size approximate kernelization scheme (PSAKS), i.e., for every \( \varepsilon > 0 \) there is a \((1 + \varepsilon)\)-approximate kernelization algorithm that computes a polynomial approximate kernel. An easy corollary then is that Steiner Tree parameterized only by \( p \) also has a PSAKS, by setting \( c = 1 \) in Theorem 5 and using the reduction from Steiner Tree to Steiner Forest as above.

**Theorem 5.** There is a \((1 + \varepsilon)\)-approximate kernelization algorithm for Steiner Forest, which given an edge-weighted undirected graph \( G = (V, E) \), a list \( \{s_1, t_1\}, \ldots, \{s_k, t_k\} \subseteq V \) of terminal pairs, and integers \( p, c \), computes an approximate kernel of size \((p + c)/\varepsilon)^{O(1/\varepsilon)}\), if for the optimum Steiner forest \( F \subseteq G \), \( p \geq |V(F) \setminus R| \) where \( R = \{s_i, t_i \mid 1 \leq i \leq k\} \), the number of connected components of \( F \) is at most \( c \), and \( \varepsilon > 0 \).\(^1\)

Analogous to approximation schemes, it is possible to distinguish between efficient and non-efficient kernelization schemes: a PSAKS is size efficient if the size of the approximate kernel is bounded by \( f(\varepsilon) \cdot p^{O(1)} \), where \( p \) is the parameter and \( f \) is a computable function independent of \( p \). Our bound on the approximate kernel size in Theorem 5 implies that we do *not* obtain a size efficient PSAKS for either Steiner Forest or Steiner Tree. This is in contrast to the existence of efficient approximation schemes for the same parameters in Theorems 1 and 4. We leave open whether a size efficient PSAKS can be found in either case. Interestingly, we also do not obtain any PSAKS for the Unweighted Directed Steiner Tree problem, even though by Theorem 2 an EPAS exists. We leave open whether a PSAKS can be found for this variant as well.

**Used techniques.** Our algorithms are based on the intuition that a Steiner tree containing only few Steiner vertices but many terminals must either contain a large component induced by terminals, or a Steiner vertex with many terminal neighbors forming a large star. A high-level description of our algorithms for Unweighted Directed Steiner Tree and Steiner Forest therefore is as follows. In each step a tree is found in the graph in polynomial time, which connects some terminals using few Steiner vertices. We save this tree as part of the approximate solution and then contract it in the graph. The vertex resulting from
the contraction is declared a terminal and the process repeats for the new graph. Previous results [24, 33] have also built on this straightforward procedure in order to obtain FPT algorithms and polynomial kernels for special cases of Unweighted Directed Steiner Tree and Unweighted Steiner Tree. In particular, in the unweighted undirected setting it is a well-known fact (cf. [33]) that contracting an adjacent pair of terminals is always a safe option, as there always exists an optimum Steiner tree containing this edge. However this immediately breaks down if the input graph is edge-weighted, as an edge between terminals might be very costly and should therefore not be contained in any (approximate) solution.

Instead we employ more subtle contraction rules, which use the following intuition. Every time we contract a tree with $\ell$ terminals we decrease the number of terminals by $\ell - 1$ (as the vertex arising from a contraction is a terminal). Our ultimate goal would be to reduce the number of terminals to one—at this point, the edges that we contracted during the whole run connect all the terminals. Decreasing the number of terminals by one can therefore be seen as a “unit of work”. We will pick a tree with the lowest cost per unit of work done, and prove that as long as there are sufficiently many terminals left in the graph, these contractions only lose an $\varepsilon$-factor compared to the optimum. As soon as the number of terminals falls below a certain threshold depending on the given parameter, we can use an FPT algorithm computing the optimum solution in the remaining graph. This algorithm is parametrized by the number of terminals, which now is bounded by our parameter. For the variants of Steiner Tree considered in our positive results, such FPT algorithms can easily be obtained from the ones for Steiner Tree [16, 2]. Adding this exact solution to the previously contracted trees gives a feasible solution that is a $(1 + \varepsilon)$-approximation.

Each step in which a tree is contracted in the graph, can be seen as a reduction rule as used for kernelization algorithms. Typically, a proof for a kernelization algorithm will define a set of reduction rules and then show that the instance resulting from applying the rules exhaustively has size bounded as a function in the parameter. To obtain an $\alpha$-approximate kernelization algorithm, additionally it is shown that each reduction rule is $\alpha$-safe. Roughly speaking, this means that at most a factor of $\alpha$ is lost when applying any number of $\alpha$-safe reduction rules.

Contracting edges in a directed graph may introduce new paths, which did not exist before. Therefore, for the Unweighted Directed Steiner Tree problem, we need to carefully choose the arborescence to contract. In order to prove Theorem 2 we show that each contraction is a $(1 + \varepsilon)$-safe reduction rule. However, the total size of the graph resulting from exhaustively applying the contractions is not necessarily bounded as a function of our parameter. Thus we do not obtain an approximate kernel.

For Steiner Forest the situation is in a sense the opposite. Choosing a tree to contract follows a fairly simple rule. On the downside however, the contractions we perform are not necessarily $(1 + \varepsilon)$-safe reduction rules. In fact there are examples in which a single contraction will lose a large factor compared to the optimum cost. We are still able to show however, that after performing all contractions exhaustively, any $\beta$-approximation to the resulting instance can be converted into a $(1 + \varepsilon)/\beta$-approximation to the original input instance. Even though the total size of the resulting instance again cannot be bounded in terms of our parameter, for Steiner Forest we can go on to obtain a PSAKS. For this we utilize a result of Lokshtanov et al. [28], which shows how to obtain a PSAKS for Steiner Tree if the parameter is the number of terminals. This result can be extended to Steiner Forest, and since our instance has a number of terminals bounded in our parameter after applying all contractions, we obtain Theorem 5.
Finally, to obtain our inapproximability result of Theorem 3, we use a reduction from the Dominating Set problem. It was recently shown by Chen and Lin [8] that this problem does not admit parameterized $\alpha$-approximation algorithms for any constant $\alpha$, if the parameter is the solution size, unless $W[1] = \text{FPT}$. We are able to exploit this to also show that no such algorithm exists for Directed Steiner Tree with edge weights, under the same assumption.

All missing proofs of this paper are deferred to the full version of the paper [17].

Related work. As the Steiner Tree problem and its variants have been studied since decades, the literature on this topic is huge. We only present a selection of related work here, that was not yet mentioned above.

For planar graphs [4] it was shown that an EPTAS exists for Steiner Tree. For Steiner Forest a 2-approximation can be computed in polynomial time on general inputs [1], but an EPTAS also exists if the input is planar [19]. If the Unweighted Steiner Tree problem is parametrized by the solution size, it is known [14] that no polynomial (exact) kernel exists, unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. If the input is restricted to planar or bounded-genus graphs it was shown that polynomial kernels do exist for this parametrization [31]. It was later shown [33] that for planar graphs this is even true for our parameter $p$. For the Directed Steiner Tree problem it is a long standing open problem whether a polylogarithmic approximation can be computed in polynomial time. It is known that an $O(|R|^2)$-approximation can be computed in polynomial time [7], and an $O(\log^2 n)$-approximation in quasi-polynomial time [7]. A recent result [21] considers generalizations of Directed Steiner Tree and characterizes which of these problems are FPT and which are $W[1]$-hard for parameter $|R|$.

2 The weighted undirected Steiner forest and Steiner tree problems

In this section we describe an approximate polynomial time preprocessing algorithm that returns an instance of Steiner Forest containing at most $O((p + c)^2/s^4)$ terminals if there is a Steiner forest with at most $p$ Steiner vertices and at most $c$ connected components. We can use this algorithm in two ways. Either we can proceed with a kernelization derived from Lokshtanov et al. [28] and obtain a polynomial size lossy kernel (Theorem 5), or we can run an exact FPT algorithm derived from Dreyfus and Wagner [16] on the reduced instance, obtaining an EPAS running in single exponential time with respect to the parameters (Theorems 1 and 4). In both cases we use the combined parameter $(p, c)$.

We first rescale all weights so that every edge has weight strictly greater than 1. Then, in each step of our algorithm we pick a star, add it to the solution, and contract the star in the current graph. We repeat this procedure until the number of terminals falls below a specified bound depending on $c$, $p$, and $c$. To describe how we pick the star to be contracted in each step, we need to introduce the ratio of a star. Let $C$ be a set of edges of a star, i.e., all edges of $C$ are incident to a common vertex which is the center of the star, and denote by $Q$ the set of terminals incident to $C$. Provided $|Q| \geq 2$, we define the ratio of $C$ as $w(C)/(|Q| - 1)$, where $w(C) = \sum_{e \in C} w(e)$. Note that we allow $C$ to contain only a single edge if it joins two terminals. Observe also that due to rescaling of edge weights each star has ratio strictly greater than 1.

In every step, our algorithm contracts a star with the best available ratio (i.e., the lowest ratio among all stars connecting at least two terminals). Due to the following lemma, a star with the best ratio has a simple form: it consists of the cheapest $i$ edges incident to its center vertex and some terminal. As there are $n$ possible center vertices and at most $n$ incident edges to each center, the best ratio star can be found in time $O(n^2)$. 
Lemma 6. Let \( v \) be a vertex and denote by \( q_1,q_2,\ldots \) the terminals adjacent to \( v \), where \( w(q_1) \leq w(q_2) \leq \cdots \), i.e., the terminals are ordered non-decreasingly by the weight of the corresponding edge \( vq_i \). The star with the best ratio having \( v \) as its center has edge set \( \{vq_1,vq_2,\ldots,vq_\ell \} \) for some \( \ell \).

To analyze our algorithm we need to keep track of the different graphs resulting from each contraction step \( t \). Initially we set \( G_0 \) to the input graph, and in each step \( t \geq 0 \) we obtain a new graph \( G_{t+1} \) from \( G_t \) by contracting a set of edges \( C_t \) in \( G_t \), such that \( C_t \) forms a star of minimum ratio in \( G_t \). That is, we obtain \( G_{t+1} \) from \( G_t \) by identifying all vertices incident to edges in \( C_t \), removing all resulting loops, and among the resulting parallel edges we delete all but the lightest one with respect to their weights. We also adjust the terminal pairs in a natural way: let \( v \) be the vertex of \( G_{t+1} \) resulting from contracting \( C_t \). If \( G_t \) had a terminal pair \( \{s,t\} \) such that \( s \) is incident to some edge of \( C_t \) while \( t \) is not, then we introduce the terminal pair \( \{v,t\} \) for \( G_{t+1} \). Also every terminal pair \( \{s,t\} \) of \( G_t \) for which neither \( s \) nor \( t \) is incident to any edge of \( C_t \) is introduced as a terminal pair of \( G_{t+1} \). Any terminal pair for which both \( s \) and \( t \) are incident to edges of \( C_t \) is going to be connected by a path in the computed solution, as it will contain \( C_t \). Hence, such a terminal pair can be safely removed.

The algorithm stops contracting best-ratio stars when there are less than \( \tau \) terminals left; the exact value of \( \tau \) depends on \( p, c, \) and the desired approximation factor and we specify it later. If this happens in step \( \ell \), the solution lifting algorithm takes a feasible solution \( F \) of \( G_\ell \) and returns the union of \( F \) and \( \bigcup_{t=0}^{\ell-1} C_t \). Such a solution is clearly feasible, since we adapted the terminal pairs accordingly after each contraction.

For the purpose of analysis, we consider a solution in the current graph \( G_t \) that originates from a solution of the original instance \( G_0 \), but may contain edges that are heavier than those in \( G_t \). More concretely, denote by \( F_0^* \) a solution in \( G_0 \) with at most \( p \) Steiner vertices and at most \( c \) components, i.e., \( F_0^* \) is a Steiner forest containing every \( s\)-\( t \) path. We remark that \( F_0^* \) may or may not be an optimum solution of \( G_0 \). Given \( F_t^* \) for \( t \geq 0 \), we modify this solution to obtain a new feasible solution \( F_{t+1}^* \) on the terminal pairs of \( G_{t+1} \). Note that the edges of the contracted star \( C_t \) might not be part of \( F_t^* \). We still mimic the contraction of the star in \( F_t^* \): to obtain \( F_{t+1}^* \) from \( F_t^* \), we identify all leaves of \( C_t \) (which are terminals by Lemma 6 and thus part of the solution \( F_t^* \)) and possibly also the center \( v \) of \( C_t \) if it is in \( F_t^* \). This results in a vertex \( v' \). We now want to delete edges incident to \( v' \) in such a way that we are left with an acyclic feasible solution. If we delete an inclusion-wise minimal feedback edge set, we clearly get a feasible solution. Let \( Q_t \) denote the set of terminals incident to \( C_t \). We choose a feedback edge set \( D_t \) for which every edge was incident to a vertex of \( Q_t \) before the contraction in \( F_t^* \), i.e., an edge of \( C_t \) corresponding to an edge of \( D_t \) never connects two Steiner vertices. Note that such an inclusion-wise minimal feedback edge set always exists: if we delete all edges of \( F_t^* \) incident to \( Q_t \) except \( C_t \) and then contract \( C_t \), we get an acyclic graph. See Figure 1 for an illustration.

The resulting graph is \( F_{t+1}^* \), which now forms a forest connecting all terminal pairs of \( G_{t+1} \). Note that for each edge in \( F_{t+1}^* \) there is a corresponding edge in \( G_{t+1} \), which however may be lighter in \( G_{t+1} \), as from each bundle of parallel edges in \( G_t \) we keep the lightest one, but this edge may not exist in \( F_t^* \).

To show that our algorithm only loses an \( \varepsilon \)-factor compared to the cost of the solution \( F_0^* \), we will compare the cost of the edges \( C_t \) contracted by our algorithm to the set \( D_t = E(F_{t+1}^*) \setminus E(F_t^*) \) of deleted edges of \( F_t^* \). Note that there are at least \( |Q_t| - c \) edges in \( D_t \), since we contracted \( Q_t \) terminals in the forest \( F_t^* \) with at most \( c \) connected components to obtain \( F_{t+1}^* \) and a forest on \( n \) vertices and \( k \) components has \( n-k \) edges. We decrease the number of vertices of \( F_t^* \) by at least \(|Q_t| - 1\) (one more if the center of the star with edge...
set $C_t$ was a Steiner vertex present in $F^*_t$), and we decrease the number of components by at most $c - 1$. Note also that for any two time steps $t \neq t'$, the sets $D_t$ and $D_{t'}$, but also the sets $C_t$ and $C_{t'}$, are disjoint. Thus if $w(C_t) \leq (1 + \varepsilon)w(D_t)$ for every $t$, then our algorithm computes a $(1 + \varepsilon)$-approximation. Unfortunately, this is not always the case: there are contractions for which this condition does not hold (see Figure 2) and we have to account for them differently.

**Definition 7.** If $w(C_t) \leq (1 + \varepsilon)w(D_t)$ we say that the contracted edge set $C_t$ in step $t$ is good; otherwise $C_t$ is bad. Moreover, if $F^*_t$ has strictly more components than $F^*_{t+1}$, we say that $C_t$ is multiple-component, otherwise it is single-component.

Our goal is to show that the total weight of bad contractions is bounded by an $\varepsilon$-fraction of the weight of $F^*_t$. We start by proving that if the set $Q_t$ of terminals in $C_t$ is sufficiently large, then the contraction is good. We define $\lambda := (1 + \varepsilon)(p + c)/\varepsilon$.

**Lemma 8.** If $|Q_t| \geq \lambda$, then the contracted edge set $C_t$ is good.

**Proof of Lemma 8.** For brevity, we drop the index $t$. Let $r = w(C)/(|Q| - 1)$ be the ratio of the contracted star, and let $\ell'$ be the number of deleted edges in $D$ that connect two terminals. Note that any such edge has weight at least $r$, since it spans a star with two terminals, which has ratio equal to its weight, and since each edge in $F^*$ (of which $D$ is a subset) can only be heavier than the corresponding edge in the current graph $G$.

Let $u_1, \ldots, u_q$ be the Steiner vertices adjacent to edges in $D$, and let $\ell_i$ be the number of edges in $D$ incident to one such Steiner vertex $u_i$. Consider the star spanned by the $\ell_i$ edges of $D$ incident to $u_i$. If $\ell_i \geq 2$, the ratio of this star is at least $r$, since its edges are at least as heavy as the corresponding edges in $G$ and the algorithm chose a star with the minimum ratio in $G$. Thus, the weight of edges in $D$ incident to $u_i$ is at least $r(\ell_i - 1)$. In the case where $\ell_i = 1$, the lower bound $r(\ell_i - 1) = 0$ on the weight holds trivially.
Any edge in $D$ not incident to any Steiner vertex $u_i$ connects two terminals. Therefore, we have $\ell' + \sum_{i=1}^{q} \ell_i = |D|$ as any edge in $D$ is incident to a terminal in $Q$ and we thus do not count any edge twice. Also recall that $|D| \geq |Q| - c$. Since $F$ contains at most $p$ Steiner vertices we have $q \leq p$, and we obtain

$$w(D) \geq r \ell' + \sum_{i=1}^{q} r(\ell_i - 1) = r \left( \ell' + \sum_{i=1}^{q} \ell_i - q \right) \geq r(|Q| - p - c).$$

Finally, using $|Q| \geq \lambda$ we bound $w(C)$ by $(1+\varepsilon)w(D)$ as follows:

$$(1+\varepsilon)w(D) \geq (1+\varepsilon)r(|Q| - p - c) = r(|Q| - 1) + r(\varepsilon|Q| - (1+\varepsilon)(p+c)+1) \geq w(C) + r \left( \frac{(1+\varepsilon)(p+c)}{\varepsilon} - (1+\varepsilon)(p+c) \right) = w(C).$$

Note that there may be a lot of contractions with $|Q| < \lambda$. However, we show that only a bounded number of them is actually bad. The key idea is to consider contractions with ratio in an interval $((1+\delta)^i; (1+\delta)^{i+1}]$ for some $\delta > 0$ and integer $i$. Due to the rescaling of weights every star belongs to an interval with $i \geq 0$. The following crucial lemma of our analysis shows that the number of bad single-component contractions in each such interval is bounded in terms of $p$ and $\varepsilon$, if $\delta$ is a function of $\varepsilon$. In particular, let $\delta := \sqrt{1+\varepsilon} - 1$, so that $(1+\delta)^2 = 1+\varepsilon$. We call an edge set $C$ with ratio $r$ in the $i$-th interval, i.e., with $r \in ((1+\delta)^i; (1+\delta)^{i+1}]$, an $i$-contraction, and define $\kappa := (1+\delta)p/\delta$.

**Lemma 9.** For any $i$ the number of bad single-component $i$-contractions is at most $\kappa$.

**Proof.** Let us focus on bad single-component $i$-contractions only which we just call bad $i$-contractions for brevity. Suppose for a contradiction that the number of bad $i$-contractions is larger than $\kappa$. Let $\tilde{t}$ be the first step with a bad $i$-contraction, i.e., $\tilde{t}$ is the minimum among all $t$ for which $w(C_t) > (1+\varepsilon)w(D_t)$ and $w(C_t)/(|Q_t| - 1) \in ((1+\delta)^i; (1+\delta)^{i+1}]$. The plan is to show that at step $\tilde{t}$ there is a “light” star in $G_{\tilde{t}}$ with ratio at most $(1+\delta)^i$ and consequently the algorithm would do a $j$-contraction for some $j < i$. This leads to a contradiction, since we assumed that in step $\tilde{t}$ the contraction has ratio in interval $i$. Note that it is sufficient to find such a light star in $F^*_t$ as for each edge in $F^*_t$ there is an edge in the graph $G_{\tilde{t}}$ between the same vertices of the same weight or even lighter.

We claim that for each step $t$ in which the algorithm does a bad $i$-contraction there is an edge $e_t \in D_t$ with weight at most $(1+\delta)^{i-1}$. We have $w(C_t) > (1+\varepsilon)w(D_t)$ as $C_t$ is bad and $w(C_t) \leq (1+\delta)^{i+1}|Q_t| - 1)$ as the ratio of $C_t$ is in interval $i$. Putting it together and using the definition of $\delta$ we obtain $w(D_t) < (1+\delta)^{i+1}/(1+\varepsilon) \cdot (|Q_t| - 1) = (1+\delta)^{i-1}|Q_t| - 1)$. Because $C_t$ is single-component, we have $|D_t| \geq |Q_t| - 1$ and therefore there is an edge $e_t \in D_t$ with weight at most $(1+\delta)^{i-1}$, which proves the claim.

Since edges between terminals have weight at least $(1+\delta)^i$ in step $t$ (recall that each such edge is a star with ratio equal to its weight), the edge $e_t$ is incident to a Steiner vertex in $F^*_t$ (otherwise the algorithm would have chosen one of the edges between terminals to contract). As $D_t \cap D_{t'} = \emptyset$, the edges $e_t$ and $e_{t'}$ for steps $t \neq t'$ with bad $i$-contractions are distinct. Let $S$ be the set of such light edges $e_t$. We have $|S| > \kappa$ as there are more than $\kappa$ bad $i$-contractions.

Since the solution $F^*_0$ uses at most $p$ Steiner vertices, also $F^*_t$ contains at most $p$ of them. Therefore at $\tilde{t}$ there must be a Steiner vertex $v$ in $F^*_\tilde{t}$ incident to at least $|S|/p > \kappa/p \geq (1+\delta)/\delta$ edges in $S$. Consider a star $C$ with $v$ as the center and with edges from $S$ that are incident to $v$; we have $|C| \geq (1+\delta)/\delta$. The ratio of this star is at most $|C|/(1+\delta)^{i-1}/(|C| - 1)$. 

\[ \text{EPAS for Steiner Trees with Small Number of Steiner Vertices} \]
Since $|C|/(|C| - 1) \leq (1 + \delta)$ (by a routine calculation) we get that the ratio of $C$ is at most $(1 + \delta)^i$ which is a contradiction to the assumption that the algorithm does an $i$-contraction in step $f$.

We also need a bound on number of bad multiple-component edge sets.

Lemma 10. The number of steps $t$ in which a bad multiple-component edge set $C_t$ is contracted is at most $c - 1$.

Proof. If $C_t$ is a bad multiple-component edge set, $F_{t+1}^*$ must have at least one component fewer than $F_t^*$. Since $F_0^*$ has at most $c$ components, the bound follows.

We remark that the proofs of Lemmas 9 and 10 do not use that the number of terminals in a bad $i$-contraction is bounded by $\lambda$, as shown in Lemma 8. Instead we bound the total weight of bad contractions in terms of $\lambda$. For this let $j$ be the largest interval of any contraction during the whole run of the algorithm, i.e., the ratio of every contracted star is at most $(1 + \delta)^{j+1}$. As there are at most $\kappa$ bad single-component contractions in each interval and $c$ bad multiple-component contractions and the interval size grows exponentially, we can upper bound the total weight of bad contractions in terms of $\kappa, c, \lambda$ and $(1 + \delta)^{j}$. We can also lower bound the weight of $w(F_{0}^*)$ in terms of $(1 + \delta)^{j}$ and the lower bound $\tau$ on the number of terminals in the graph. If $\tau$ is large enough then the total weight of edge sets $C_t$ of bad contractions is at most $\varepsilon \cdot w(F_{0}^*)$. These ideas are summarized in the next lemma.

Lemma 11. Let $j$ be the largest interval of any contraction during the whole run of the algorithm and let $W_B$ be the total weight of edge sets $C_t$ of bad contractions. Then, the following holds.
1. $W_B \leq (\kappa + c) \cdot \lambda \cdot (1 + \delta)^{j+2}/\delta$.
2. $w(F_{0}^*) \geq (1 + \delta)^{j} \cdot (\tau - 2p - c)$.
3. If $\tau := (\kappa + c) \cdot \lambda \cdot (1 + \delta)^{j}/(\varepsilon \delta) + 2p + c$ then $W_B \leq \varepsilon \cdot w(F_{0}^*)$.

The above lemma can now be used to prove that all the contractions put together (by scaling $\varepsilon$) form a $(1 + \varepsilon)$-approximate pre-processing procedure with respect to $F_{0}^*$.

Lemma 12. The algorithm outputs an instance with $\tau \in O((p + c)^2/\varepsilon^4)$ terminals and (together with the solution lifting algorithm) it is a $(1 + 2\varepsilon)$-approximate polynomial time pre-processing algorithm with respect to $F_{0}^*$.

Note that in case the given $p$ is smaller than the number of Steiner vertices in $F_{0}^*$, or $c$ is smaller than the number of connected components in $F_{0}^*$, the algorithm still outputs a Steiner forest, but the approximation factor may be arbitrary. The last lemma can easily be used to prove Theorems 1, 4 and 5.

3 The unweighted directed Steiner tree problem

In this section we provide an EPAS for the UNWEIGHTED DIRECTED STEINER TREE problem, in which each arc has unit weight. The idea behind our algorithm given in this section is to reduce the number of terminals of the input instance via a set of reduction rules. That is, we would like to reduce the input graph $G$ to a graph $H$, and prove that the number of terminals in $H$ is bounded by a function of our parameter $p$ and the desired approximation ratio. On $H$ we use the algorithm of Björklund et al. [2] to obtain an optimum solution.

Our first reduction rule represents the idea that a terminal in the immediate neighborhood of the root can be contracted to the root. Observe that in this case our algorithm has to
pay 1 for connecting such a terminal to the root, however, any feasible solution must connect this terminal as well using at least one arc—this argument is formalized in Lemma 13.

▶ Reduction Rule R1. If there is an arc from the root \( r \) to a terminal \( v \in R \), we contract the arc \((r,v)\), and declare the resulting vertex the new root.

▶ Lemma 13. Reduction Rule R1 is 1-safe and can be implemented in polynomial time. Furthermore, there is a solution lifting algorithm running in polynomial time and returning a Steiner arborescence if it gets a Steiner arborescence of the reduced graph as input.

The idea behind our next reduction rule is the following. Assume there is a Steiner vertex \( s \) in the optimum arborescence \( T \) connected to many terminals with paths not containing any other Steiner vertices. We can then afford to buy all these paths emanating from \( s \) together with a path connecting the root to \( s \). Formally, we say that a vertex \( u \) is a \( k \)-extended neighbor of some vertex \( v \), if there exists a directed path \( P \) starting in \( v \) and ending in \( u \), such that \( V(P) \setminus \{v\} \) contains at most \( k \) Steiner vertices. Note that a vertex is always a \( k \)-extended neighbor of itself for any \( k \), and that each of the above terminals connected to \( s \) in \( T \) is a 0-extended neighbor of \( s \). We denote by \( N^k_{\text{Ext}}(v) \) the set of all \( k \)-extended neighbors of \( v \), and call it the \( k \)-extended neighborhood of \( v \) (see Figure 3). By the following observation the Steiner vertex \( s \) of \( T \) lies in the \( p \)-extended neighborhood of the root \( r \). Therefore there is a path containing at most \( p \) Steiner vertices connecting \( r \) to \( s \).

▶ Observation 14. Let \( G = (V,A) \) be a directed graph with root \( r \in R \). Suppose there exists a Steiner arborescence \( T \subseteq G \) with at most \( p \) Steiner vertices. It follows that \( V(T) \subseteq N^p_{\text{Ext}}(r) \).

In what follows we fix \( \varepsilon > 0 \). The second reduction rule contracts a path from \( r \) to a Steiner vertex \( s \) in the \( p \)-extended neighborhood of \( r \) together with the 0-extended neighborhood of \( s \) if this neighborhood is sufficiently large.

▶ Reduction Rule R2. If there exists a Steiner vertex \( s \) with \( |N^0_{\text{Ext}}(s)| \geq \frac{p}{\varepsilon} \) and \( s \in N^p_{\text{Ext}}(r) \), so that there is an \( r \to s \) path \( P \) containing at most \( p \) Steiner vertices, then we contract the subgraph of \( G \) induced by \( N^0_{\text{Ext}}(s) \) and \( P \) in \( G \), and declare the resulting vertex the new root.

▶ Lemma 15. Reduction Rule R2 is \( (1 + \varepsilon) \)-safe and can be implemented in polynomial time. Furthermore, there is a solution lifting algorithm running in polynomial time and returning a Steiner arborescence if it gets a Steiner arborescence of the reduced graph as input.

Now we prove that if none of the above reduction rules is applicable and our algorithm was provided with a correct value for parameter \( p \), then the number of terminals in the reduced graph can be bounded by \( \frac{p^2}{\varepsilon} \).

▶ Lemma 16. Let \( G \) be an instance of Directed Steiner Tree, and denote by \( H \) the graph obtained from \( G \) by exhaustive application of Reduction Rules R1 and R2. Suppose
that there exists a Steiner arborescence in $G$ containing at most $p$ Steiner vertices. It follows that the remaining terminal set $R$ of $H$ has size less than $p^2/\varepsilon$.

The last step of the algorithm is to compute an optimum solution in the graph $H$ obtained from the input graph $G$ after exhaustively applying the two above reduction rules. From the resulting arborescence in $H$ we obtain an arborescence in $G$ by running the solution lifting algorithms for each reduction rule applied (in the reverse order); the existence and correctness of the solution lifting algorithms for our reduction rules is provided by Lemmas 13 and 15.

---

**References**


EPAS for Steiner Trees with Small Number of Steiner Vertices


Fixed Parameter Approximations for $k$-Center Problems in Low Highway Dimension Graphs

Andreas Emil Feldmann

Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada
andreas.feldmann@uwaterloo.ca

Abstract. We consider the $k$-Center problem and some generalizations. For $k$-Center a set of $k$ center vertices needs to be found in a graph $G$ with edge lengths, such that the distance from any vertex of $G$ to its nearest center is minimized. This problem naturally occurs in transportation networks, and therefore we model the inputs as graphs with bounded highway dimension, as proposed by Abraham et al. [ICALP 2011].

We show both approximation and fixed-parameter hardness results, and how to overcome them using fixed-parameter approximations. In particular, we prove that for any $\varepsilon > 0$ computing a $(2-\varepsilon)$-approximation is $\mathsf{W}[2]$-hard for parameter $k$, and NP-hard for graphs with highway dimension $O(\log^2 n)$. The latter does not rule out fixed-parameter $(2-\varepsilon)$-approximations for the highway dimension parameter $h$, but implies that such an algorithm must have at least doubly exponential running time in $h$ if it exists, unless the ETH fails. On the positive side, we show how to get below the approximation factor of 2 by combining the parameters $k$ and $h$: we develop a fixed-parameter $3/2$-approximation with running time $2^{O(kh \log h)} \cdot n^{O(1)}$.

We also provide similar fixed-parameter approximations for the weighted $k$-Center and $(k, F)$-Partition problems, which generalize $k$-Center.

1 Introduction

In this paper we consider the $k$-Center problem and some of its generalizations. For the problem, $k$ locations need to be found in a network, so that every node in the network is close to a location. More formally, the input is specified by an integer $k \in \mathbb{N}$ and a graph $G = (V, E)$ with positive edge lengths. A feasible solution to the problem is a set $C \subseteq V$ of centers such that $|C| \leq k$. The aim is to minimize the maximum distance between any vertex and its closest center. That is, let $\text{dist}_G(u, v)$ denote the shortest-path distance between two vertices $u, v \in V$ of $G$ according to the edge lengths, and $B_v(r) = \{u \in V \mid \text{dist}_G(u, v) \leq r\}$ be the ball of radius $r$ around $v$. We need to minimize the cost of the solution $C$, which is the smallest value $\rho$ for which $\bigcup_{v \in C} B_v(\rho) = V$. We say that a center $v \in C$ covers a vertex $u \in V$ if $u \in B_v(\rho)$. Hence we can see the problem as finding $k$ centers covering all vertices of $G$ with balls of minimum radius.

I would like to thank Jochen Könemann for reading a draft of this paper. This work was supported by ERC Starting Grant PARAMTIGHT (No. 280152).
The $k$-Center problem naturally arises in transportation networks, where for instance it models the need to find locations for manufacturing plants, hospitals, police stations, or warehouses under a budget constraint. Unfortunately it is NP-hard to solve the problem in general [23], and the same holds true in various models for transportation networks, such as planar graphs [22] and metrics using Euclidean ($L_2$) or Manhattan ($L_1$) distance measures [13]. A more recent model for transportation networks uses the highway dimension, which was introduced as a graph parameter by Abraham et al. [1]. The intuition behind its definition comes from the empirical observation [6,7] that in a road network, starting from any point $A$ and travelling to a sufficiently far point $B$ along the quickest route, one is bound to cross one of relatively few “access points”. There are several formal definitions for the highway dimension that differ slightly [1,3,2]. All of them however imply the existence of locally sparse shortest path covers. Therefore, in this paper we consider this as a generalization of the original highway dimension definitions (in fact the definition given in [2] is equivalent to this).

**Definition 1.** Given a graph $G = (V, E)$ with edge lengths and a scale $r \in \mathbb{R}^+$, let $P_{(r,2r)} \subseteq 2^V$ contain all vertex sets given by shortest paths in $G$ of length more than $r$ and at most $2r$. A shortest path cover $spc(r) \subseteq V$ is a hitting set for the set system $P_{(r,2r)}$, i.e. $P \cap spc(r) \neq \emptyset$ for each $P \in P_{(r,2r)}$. We call the vertices in $spc(r)$ hubs. A hub set $spc(r)$ is called locally $h$-sparse, if for every vertex $v \in V$ the ball $B_v(2r)$ of radius $2r$ around $v$ contains at most $h$ vertices from $spc(r)$. The highway dimension of $G$ is the smallest integer $h$ such that there is a locally $h$-sparse shortest path cover $spc(r)$ for every scale $r \in \mathbb{R}^+$ in $G$.

Abraham et al. [1] introduced the highway dimension in order to explain the fast running times of various shortest-path heuristics. However they also note that “conceivably, better algorithms for other [optimization] problems can be developed and analysed under the small highway dimension assumption”. In this paper we investigate the $k$-Center problem and focus on graphs with low highway dimension as a model for transportation networks. One advantage of using such graphs is that they do not only capture road networks but also networks with transportation links given by air-traffic or railroads. For instance, introducing connections due to airplane traffic will render a network non-planar, while it can still be argued to have low highway dimension. This is because longer flight connections tend to be serviced by bigger but sparser airports, which act as hubs. This can for instance be of interest in applications where warehouses need to be placed to store and redistribute goods of globally operating enterprises. Unfortunately however, we show in this paper that the $k$-Center problem also remains NP-hard on graphs with low highway dimension.

Two popular and well-studied ways of coping with NP-hard problems is to devise approximation [23] and fixed-parameter [12] algorithms. For the former we demand polynomial running times but allow the computed solution to deviate from the optimum cost. That is, we compute a $c$-approximation, which is a feasible solution with a cost that is at most $c$ times worse than the best possible for the given instance. A problem that allows a polynomial-time $c$-approximation for
any input is $c$-approximable, and $c$ is called the approximation factor of the corresponding algorithm. The rational behind fixed-parameter algorithms is that some parameter $p$ of the input is small and we can therefore afford running times that are super-polynomial in $p$, where however we demand optimum solutions. That is, we compute a solution with optimum cost in time $f(p) \cdot n^{O(1)}$ for some function $f(\cdot)$ that is independent of the input size $n$. A problem that has a fixed-parameter algorithm for a parameter $p$ is called fixed-parameter tractable (FPT) for $p$. What however, if a problem is neither approximable nor FPT? In this case it may be possible to overcome the complexity by combining these two paradigms. In particular, the objective becomes to develop fixed-parameter $c$-approximation (c-FPA) algorithms that compute a $c$-approximation in time $f(p) \cdot n^{O(1)}$ for a parameter $p$.

The idea of combining the two paradigms of approximation and fixed-parameter tractability has been suggested before. However only few results are known for this setting (cf. [21]). In this paper we show that for the $k$-Center problem it is possible to overcome lower bounds for its approximability and its fixed-parameter tractability using fixed-parameter approximations. For many different input classes, such as planar graphs [22], and $L_2$- and $L_\infty$-metrics [13], the $k$-Center problem is 2-approximable but not $(2 - \varepsilon)$-approximable for any $\varepsilon > 0$, unless P=NP. We show that, unless P=W[2], for general graphs there is no $(2 - \varepsilon)$-FPA algorithm for the parameter $k$. Additionally, we prove that, unless P=NP, $k$-Center is not $(2 - \varepsilon)$-approximable on graphs with highway dimension $O(\log^2 n)$. This does not rule out $(2 - \varepsilon)$-FPA algorithms for the highway dimension parameter, and we leave this as an open problem. However the result implies that if such an algorithm exists then its running time must be enormous. In particular, unless the exponential time hypothesis (ETH) fails, there can be no $(2 - \varepsilon)$-FPA algorithm with doubly exponential $2^{2^{o(\sqrt{h})}} \cdot n^{O(1)}$ running time in the highway dimension $h$.

In face of these hardness results, it seems tough to beat the approximation factor of 2 for $k$-Center, even when considering fixed-parameter approximations for either the parameter $k$ or the highway dimension. Our main result however is that we can obtain a significantly better approximation factor for $k$-Center when combining these two parameters. Such an algorithm is useful when aiming for high quality solutions, for instance in a setting where only few warehouses should be built in a transportation network, since warehouses are expensive or stored goods should not be too dispersed for logistical reasons.

It is known [2] that locally $O(h \log h)$-sparse shortest path covers can be computed for graphs of highway dimension $h$ in polynomial time. In the following theorem summarizing our main result, the first given running time assumes this approximation. In general it is NP-hard to compute the highway dimension [14], but it is unknown whether this problem is FPT. If this is the case and the running time is sufficiently small, this can be used as an oracle in our algorithm.

**Theorem 2.** For any graph $G$ with $n$ vertices and of highway dimension $h$, there is an algorithm that computes a $3/2$-approximation to the $k$-Center problem in time $2^{O(kh \log h)} \cdot n^{O(1)}$. If locally $h$-sparse shortest path covers are given by an oracle the running time is $9^{kh} \cdot n^{O(1)}$. 
We leave open whether approximation factors better than $3/2$ can be obtained for the combined parameter $(k, h)$. Even if we also leave open whether $(2 - \varepsilon)$-FPA algorithms exist for the parameter $h$ alone, we are able to prove that the techniques we use to obtain Theorem 2 cannot omit using both $k$ and $h$ as parameters. To obtain a $(2 - \varepsilon)$-FPA algorithm with running time $f(h) \cdot n^{O(1)}$ for any function $f(\cdot)$ independent of $n$, a lot more information of the input would need to be exploited than the algorithm of Theorem 2 does. To explain this, we now turn to the used techniques.

1.1 Used Techniques

A crucial observation for our algorithm is that at any scale $r$, a graph of low highway dimension is structured in the following way (Figure 1). We will prove that the vertices are either at distance at most $r$ from some hub, or they lie in clusters of diameter at most $r$ that are at distance more than $2r$ from each other. Hence, for the cost $\rho$ of the optimum solution, at scale $r = \rho/2$ a center that resides in a cluster cannot cover any vertices of some other cluster. In this sense the clusters are “independent” of each other. At the same time we are able to bound the number of hubs of scale $\rho/2$ in terms of $k$ and the highway dimension. Intuitively, this is comparable to graphs with small vertex cover, since the vertices that are not part of the vertex cover form an independent set. In this sense the highway dimension is a generalization of the vertex cover number (this is in fact the reason why computing the highway dimension is NP-hard [14]).

At the same time the $k$-Center problem is a generalization of the Dominating Set problem. This problem is W[2]-hard [12], which, as we will show, is also why $k$-Center is W[2]-hard to approximate for parameter $k$. However Dominating Set is FPT using the vertex cover number as the parameter [5]. This is one of the reasons why combining the two parameters $k$ and $h$ yields a 3/2-FPA algorithm for $k$-Center. In fact the similarity seems so striking at first that one is tempted to either reduce the problem of finding a 3/2-approximation for $k$-Center on low highway dimension graphs to solving Dominating Set on a graph of low vertex cover number, or at least use the known techniques for the latter problem to solve the former. However it is unclear how this can be made to work. Instead we devise a more involved algorithm that is driven by the intuition that the two problems are similar. The intuition works on two levels. The first part of the algorithm will determine some of the approximate centers by exploiting the fact that a center in a cluster cannot cover vertices of other clusters. In the second part of our algorithm we will actually reduce the problem of finding the remaining approximate centers to Dominating Set in a graph with small vertex cover number. Proving that the found centers form a feasible approximate solution needs a non-trivial proof.

The algorithm will guess the cost $\rho$ of the optimum solution in order to exploit the structure of the graph given by the locally $h$-sparse shortest path.
cover for scale $r = \rho/2$. In particular, the shortest path covers of other scales do not need to be locally sparse in order for the algorithm to succeed. We will show that there are graphs for which $k$-Center is not $(2-\varepsilon)$-approximable, unless $P=NP$, and for which the shortest path cover for scale $\rho/2$ is locally 25-sparse. Hence our techniques, which only consider the shortest path cover of scale $\rho/2$, cannot yield a $(2-\varepsilon)$-FPA algorithm for parameter $h$. The catch is though that the reduction produces graphs which do not have locally sparse shortest path covers for scales significantly larger than $\rho/2$. Hence a $(2-\varepsilon)$-FPA algorithm for parameter $h$ might still exist. However such an algorithm would have to take larger scales into account than just $\rho/2$, and as argued above, it would have to have at least doubly exponential running time in $h$.

Proving that no $(2-\varepsilon)$-FPA algorithm for parameter $k$ exists for $k$-Center, unless $P=W[2]$, is straightforward given the original reduction of Hsu and Nemhauser [19] from the $W[2]$-hard Dominating Set problem. For parameter $h$ however we develop some more advanced techniques. For the reduction we show how to construct a graph of low highway dimension given a metric of low doubling dimension, so that distances between vertices are preserved by a $(1+\varepsilon)$ factor. The doubling dimension [16] is a parameter that captures the bounded volume growth of metrics, such as given by Euclidean and Manhattan distances. Formally, a metric has doubling dimension $d$ if for every $r \in \mathbb{R}^+$ and vertex $v$, the ball $B_v(2r)$ of radius $2r$ is the union of at most $2^d$ balls of radius $r$. Since $k$-Center is not $(2-\varepsilon)$-approximable in $L_\infty$-metrics [13], unless $P=NP$, and these have doubling dimension 2, we are able to conclude that the hardness translates to graphs of highway dimension $O(\log^2 n)$.

1.2 Generalizations

In addition to $k$-Center, we can obtain similar results for two generalizations of the problem by appropriately modifying our techniques. For the weighted $k$-Center problem, the vertices have integer weights and the objective is to choose centers of total weight at most $k$ to cover all vertices with balls of minimum radius (see [23] for a formal definition). This problem is 3-approximable [18,23] and no better approximation factor is known. However we are able to modify our techniques to obtain a 2-FPA algorithm for the combined parameter $(k, h)$.

An alternative way to define the $k$-Center problem is in terms of finding a star cover of size $k$ in a metric, where the cost of the solution is the longest of any star edge in the solution. More generally, in their seminal work Hochbaum and Shmoys [18] defined the $(k, F)$-Partition problem (see [18] for a formal definition). Here a family of (unweighted) graphs $F$ is given and the aim is to partition the vertices of a metric into $k$ sets and connect the vertices of each set by a graph from the family $F$. The solution cost is measured by the “bottleneck”, which is the longest distance between any two vertices of the metric that are connected by an edge in a graph from the family $F$. The case when $F$ contains only stars

---

1 The details of the generalizations, together with all missing proofs of this paper, are deferred to the full version.
is exactly the $k$-Center problem, given the shortest-path metric as input. The $(k, \mathcal{F})$-Partition problem is $2d$-approximable [18], where $d$ is the largest diameter of any graph in $\mathcal{F}$. We show that a $3\delta$-FPA algorithm for the combined parameter $(k, h)$ exists, where $\delta$ is the largest radius of any graph in $\mathcal{F}$. Hence for graph families in which $3\delta < 2d$ this improves on the general algorithm by Hochbaum and Shmoys [18]. This is for example the case when $\mathcal{F}$ contains “stars of paths”, i.e. stars for which each edge is replaced by a path of length $\delta$. The diameter of such a graph is $2\delta$, while the radius is $\delta$, and hence $3\delta < 2d = 4\delta$.

### 1.3 Related Work

Given its applicability to various problems in transportation networks, but also in other contexts such as image processing and data-compression, the $k$-Center problem has been extensively studied in the past. We only mention closely related results here, that were not mentioned before. For planar and map graphs the $k$-Center problem is FPT [11] for the combined parameter $(k, \rho)$. Note though that $k$ and $\rho$ are somewhat opposing parameters in the sense that typically if $k$ is small then $\rho$ will be large, and vice versa. It would therefore be interesting to know if there are $(2 - \varepsilon)$-FPA algorithms for $k$-Center on planar or map graphs that do not use $\rho$ as a parameter. For metrics with Euclidean or Manhattan distances, $(1 + \varepsilon)$-FPA algorithms for the combined parameter $(k, \varepsilon, D)$ can be obtained [4,17], where $D$ is the dimension of the geometric space.

Generally, fixed-parameter approximations have not been intensively studied so far. A survey is given by Marx [21]. Some newer developments include $(1 + \varepsilon)$-FPA algorithms [20] for problems such as Max Cut, Edge Dominating Set, or Graph Balancing, for parameters such as treewidth and cliquewidth combined with $\varepsilon$. In terms of lower bounds, Bonnet et al. [9] make a connection between the linear PCP conjecture and fixed-parameter inapproximability for problems such as Independent Set.

Abraham et al. [1] introduce the highway dimension, and study it in various papers [1,3,2]. Their main interest is in explaining the good performance of various shortest-path heuristics assuming low highway dimension. In [2] they show that a locally $O(h \log h)$-sparse shortest path cover can be computed in polynomial time for any scale if the highway dimension of the input graph is $h$, and each shortest path is unique. We will assume that the latter is always the case, since we can slightly perturb the edge lengths. Feldmann et al. [14] consider computing approximations for various other problems that naturally arise in transportation networks. They show that quasi-polynomial time approximation schemes can be obtained for problems such as Travelling Salesman, Steiner Tree, or Facility Location, if the highway dimension is constant. This is done by probabilistically embedding a low highway dimension graph into a bounded treewidth graph while introducing arbitrarily small distortions of distances. Known algorithms to compute optimum solutions on low treewidth graphs then imply the approximation schemes. It is interesting to note that this approach does not work for the $k$-Center problem since, in contrast to the above mentioned problems, its objective function is not linear in the edge lengths. The only other theoretical
result mentioning the highway dimension that we are aware of is by Bauer et al. [8], who show that for any graph \( G \) there exist edge lengths such that the highway dimension is \( \Omega(\text{pw}(G)/\log n) \), where \( \text{pw}(G) \) is the pathwidth of \( G \).

2 Highway Dimension and Vertex Covers

We observe that the vertices of a low highway dimension graph are highly structured for any scale \( r \): the vertices that are far from any hub of a shortest path cover for scale \( r \) are clustered into sets of small diameter and large inter-cluster distance (Figure 1). This observation was already made in [14]. We first formally define the clusters and then prove that they have the claimed properties. For a set \( S \subseteq V \) let \( \text{dist}_G(u, S) = \min_{v \in S} \text{dist}_G(u, v) \) be the distance from \( u \) to the closest vertex in \( S \).

**Definition 3.** Fix \( r \in \mathbb{R}^+ \) and \( \text{spc}(r) \subseteq V \) in a graph \( G = (V, E) \). We call an inclusion-wise maximal set \( T \subseteq \{v \in V \mid \text{dist}_G(v, \text{spc}(r)) > r\} \) with \( \text{dist}_G(u, v) \leq r \) for all \( u, v \in T \) a cluster, and we denote the set of all clusters by \( \mathcal{T} \). The non-cluster vertices are those which are not contained in any cluster of \( \mathcal{T} \).

Note that the set \( \mathcal{T} \) is specific for the scale \( r \) and the hub set \( \text{spc}(r) \). The following lemma characterizes the structure of the clusters and non-cluster vertices. Here we let \( \text{dist}_G(S, S') = \min_{v \in S} \text{dist}_G(v, S') \) be the minimum distance between vertices of two sets \( S \) and \( S' \).

**Lemma 4.** Let \( \mathcal{T} \) be the cluster set for a scale \( r \) and a shortest path cover \( \text{spc}(r) \). For each non-cluster vertex \( v \) there is a hub in \( \text{spc}(r) \) with \( \text{dist}_G(u, v) \leq r \) for all \( u, v \in T \) a cluster, and we denote the set of all clusters by \( \mathcal{T} \). The diameter of any cluster \( T \in \mathcal{T} \) is at most \( r \), and \( \text{dist}_G(T, T') > 2r \) for any distinct pair of clusters \( T, T' \in \mathcal{T} \).

Intuitively, Lemma 4 means that a low highway dimension graph has similarities to a graph with small vertex cover. For this reason, as part of our algorithm we compute the optimum dominating set of a graph with a small vertex cover. To show how this can be done, we begin by formally defining the relevant notions. Let \( G = (V, E) \) be an (unweighted) graph. A dominating set \( D \subseteq V \) is a set of vertices such that every vertex of \( V \) is adjacent to some vertex of \( D \). The Dominating Set problem is to find a dominating set of minimum size. A vertex cover \( W \subseteq V \) is a set of vertices such that every edge of \( E \) is incident to some vertex of \( W \). The vertex cover number is the size of the smallest vertex cover. The reason why the Dominating Set problem is FPT for this parameter essentially is that the vertex cover number is an upper bound on the pathwidth [15]. For the following lemma, we simplify an algorithm due to Alber et al. [5] to solve Dominating Set on bounded treewidth graphs. This improves the running time. Note that the algorithm assumes the vertex cover to be given explicitly.

**Lemma 5.** Given a graph \( G \) and a vertex cover \( W \) of \( G \), the Dominating Set problem can be solved in time \( O(3^l \cdot n^2) \), where \( l = |W| \).
3 The Fixed-Parameter Approximation Algorithm

If the optimum cost for $k$-Center is $\rho$ and there is a locally $s$-sparse shortest path cover $\text{spc}(r)$ for scale $r = \rho/2$, then $|\text{spc}(r)| \leq ks$. This is because there are $k$ balls of radius $\rho$ covering the whole graph, and by Definition 1 there are at most $s$ hubs in each ball. If the input graph has highway dimension $h$ and there is an oracle that gives locally $h$-sparse shortest path covers for each scale, then we can set $s = h$. Otherwise, Abraham et al. [2] show how to compute $O(\log h)$-approximations of shortest path covers in polynomial time, if shortest paths have unique lengths. The latter can be assumed by perturbing the edge lengths and therefore we can set $s = O(h \log h)$.

The hubs of $\text{spc}(r)$ hit all shortest paths of length in $(r, 2r]$ in the same way a vertex cover hits all single edges, which can be thought of as unit length paths in an unweighted graph. Furthermore, we know that the clusters are all more than $2r = \rho$ apart by Lemma 4. Hence any center of the optimum solution to $k$-Center that lies in a cluster cannot cover any vertex of another cluster. Compared to the Dominating set problem, this is analogous to the vertices of a graph that do not belong to the vertex cover and therefore form an independent set: a vertex of an optimum dominating set that is part of the independent set cannot dominate another vertex of the independent set.

The first part of our algorithm is driven by this intuition. After guessing the optimum cost $\rho$ and computing $\text{spc}(\rho/2)$ together with its cluster set $T$, we will see how the algorithm computes three approximate center sets. For the first set $C_1$ the algorithm guesses a subset of the hubs of $\text{spc}(\rho/2)$ that are close to the optimum center set. This can be done in exponential time in $ks$ because there are at most that many hubs. The independence property of the clusters then makes it easy to determine a second set $C_2$ of approximate centers, each of which lies in a cluster that must contain an optimum center. To determine the third set of centers $C_3$, the similarity of our problem to Dominating Set on graphs with small vertex cover number becomes more concrete in our algorithm: we reduce finding $C_3$ to Dominating Set in such a way that the constructed graph has a vertex cover number that can be bounded in the number of hubs of $\text{spc}(\rho/2)$.

More concretely, consider an input graph $G = (V, E)$ with an optimum $k$-Center solution $C^*$ with cost $\rho$. For an index $i \in \{1, 2, 3\}$ we denote by $R_i^* = \bigcup_{v \in C_i^*} B_v(\rho)$ and $R_i = \bigcup_{v \in C_i} B_v(\frac{3}{2} \rho)$ the regions covered by some set of optimum centers $C_i^* \subseteq C^*$ (with balls of radius $\rho$) and approximate centers $C_i \subseteq V$ (with balls of radius $\frac{3}{2} \rho$), respectively. The algorithm tries every scale $r$ in order to guess the correct value for which $r = \rho/2$. After computing $\text{spc}(r)$ together with its cluster set $T$, the algorithm first checks that the number of hubs is not too large, in order to keep the running time low. In particular, since we know that $|\text{spc}(\rho/2)| \leq ks$, we can dismiss any shortest path cover containing more hubs. Assume that $r = \rho/2$ was found. The next step is to guess a minimal set $H$ of hubs in $\text{spc}(\rho/2)$, such that the balls of radius $\rho/2$ around hubs in $H$ cover all optimum non-cluster centers. That is, if $C_1^* \subseteq C^*$ denotes the set of optimum centers which each are at distance at most $\rho/2$ from some hub in $\text{spc}(\rho/2)$, then $C_1^* \subseteq \bigcup_{v \in H} B_v(\rho/2)$ and $H$ is minimal with this property.
We choose this set of hubs $H$ as the first set of centers $C_1$ for our approximate solution. Note that due to the minimality of $H$, $|C_1| \leq |C_1^*|$. Also $R_1^* \subseteq R_1$ since for any center in $C_1^*$ there is a center at distance at most $\rho/2$ in $C_1$.

The next step is to compute a set of centers so that all clusters of the cluster set $T$ of $\text{spc}(\rho/2)$ are covered. Some of the clusters are already covered by the first set of centers $C_1$, and thus in this step we want to cover all clusters in $U = \{ T \in T \mid T \setminus R_1 \neq \emptyset \}$. By the definition of $C_1^*$, any remaining optimum center in $C_1^* \setminus C_1^*$ must lie in a cluster. Furthermore, the distance between clusters of $\text{spc}(\rho/2)$ is more than $\rho$ by Lemma 4, so that a center of $C_1^* \setminus C_1^*$ in a cluster $T$ cannot cover any vertices of another cluster $T' \neq T$. Hence if we guessed $H$ correctly we can be sure that each cluster $T \in U$ must contain a center of $C_1^* \setminus C_1^*$. For each such cluster we pick an arbitrary vertex $v \in T$ and declare it a center of the second set $C_2$ for our approximate solution. Thus if the optimum set of centers for $U$ is $C_2^* = \{ v \in C_1^* \mid \exists T \in U : v \in T \}$, we have $|C_2| \leq |C_2^*|$. Moreover, since the diameter of each cluster is at most $\rho/2$ by Lemma 4, we get $R_2^* \subseteq R_2$.

At this time we know that all clusters in $T$ are covered by the region $R_1 \cup R_2$. Hence if any uncovered vertices remain in $V \setminus (R_1 \cup R_2)$ for our current approximate solution, they must be non-cluster vertices. By our definition of $C_1^*$ and $C_2^*$, all remaining optimum centers $C_3^* = C_1^* \setminus (C_1^* \cup C_2^*)$ lie in clusters of $T \setminus U$. Since $R_1^* \subseteq R_1$ and $R_2^* \subseteq R_2$, any remaining vertex of $V \setminus (R_1 \cup R_2)$ must be in the region $R_3^*$ covered by centers in $C_3^*$. Next we show how to compute a set $C_3$ such that the region $R_3$ includes all remaining vertices of the graph and $|C_3| \leq |C_3^*|$. Note that the latter means that the number of centers in $C_1 \cup C_2 \cup C_3$ is at most $k$, since $C_1^*$, $C_2^*$, and $C_3^*$ are disjoint.

To control the size of $C_3$ we will compute the smallest number of centers that cover parts of $R_3^*$ with balls of radius $\rho$. In particular, we will guess the set of hubs $H' \subseteq \text{spc}(\rho/2) \setminus H$ that lie in the region $R_3^*$ (note that we exclude hubs of $H$ from this set). We then compute a center set $C_3$ of minimum size such that $H' \subseteq \bigcup_{v \in C_3} B_v(\rho)$. For this we reduce the problem of computing centers covering $H'$ to the Dominating Set problem in a graph of fixed vertex cover number. The reduction follows the lines of the bipartite reduction from Set Cover to Dominating Set [10]. More concretely, let $W = \bigcup_{T \in T \setminus U} T$ denote the vertices in the clusters that include $C_3^*$. For each vertex $w \in W$ we encode the set of vertices of $H'$ that it could cover with a ball of radius $\rho$, in an instance $G'$ of Dominating Set as follows. The vertices of $G'$ include all vertices of $W$ and $H'$, and we introduce an edge between a vertex $w \in W$ and $u \in H'$ if the distance between $w$ and $u$ is at most $\rho$. Hence a vertex $w \in W$ will dominate exactly the vertices of $H'$ in the ball $B_w(\rho)$. Since we are only interested in covering vertices of $H'$, additionally we introduce apex vertices $a$ and $a'$ to $G'$, together with an edge between every $w \in W$ and $a$, and an edge between $a$ and $a'$. This way all vertices of $W$ can be dominated by only $a$, and one of the apexes must be part of any dominating set in $G'$. Note that by our choice of the edges, $H' \cup \{a\}$ is a vertex cover for $G'$ and so its vertex cover number is at most $ks + 1$, as required.
If the optimum dominating set of $G'$ is $D$, we let $C_3 = D \setminus \{a, a'\}$. The following lemma shows that the size of $C_3$ is as required, and the centers of $C_3$ cover all hubs of $H'$ with balls of radius $\rho$.

**Lemma 6.** Assuming that the algorithm guessed the correct scale $r = \rho/2$ and the correct sets $H$ and $H'$, the set $C_3 = D \setminus \{a, a'\}$ is of size at most $|C_3^*|$ and $H' \subseteq \bigcup_{v \in C_3} B_v(\rho)$.

It remains to show that the three computed center sets $C_1$, $C_2$, and $C_3$ cover all vertices of $G$, which we do in the following lemma.

**Lemma 7.** Assuming that the algorithm guessed the correct scale $r = \rho/2$ and the correct sets $H$ and $H'$, the approximate center sets $C_1$, $C_2$, and $C_3$ cover all of $G$, i.e. $R_1 \cup R_2 \cup R_3 = V$.

*Proof.* The proof is by contradiction. Assume there is a $v \in V \setminus (R_1 \cup R_2 \cup R_3)$ that is not covered by the computed approximate center sets. The idea is to identify a hub $y \in \text{spc}(\rho/2)$ on the shortest path between $v$ and an optimum center $w \in C^*$ covering $v$. We will show that this hub $y$ must however be in $H'$ and therefore $v$ is in fact in $R_3$, since $v$ also turns out to be close to $y$.

To show the existence of $y$, we begin by arguing that the closest hub $x \in \text{spc}(\rho/2)$ to $v$ is neither in $H$ nor in $H'$. We know that each cluster of $T$ is in $R_1 \cup R_2$, so that $v \notin R_1 \cup R_2$ must be a non-cluster vertex. Thus by Lemma 4, $\text{dist}_G(v, x) \leq \rho/2$. The region $R_1$ in particular contains all vertices that are at distance at most $\rho/2$ from any hub in $H = C_1$. Since $v \notin R_1$ and $\text{dist}_G(v, x) \leq \rho/2$, this means that $x \notin H$. From $v \notin R_3$ we can also conclude that $x \notin H'$ as follows. By Lemma 6, $C_3$ covers all hubs of $H'$ with balls of radius $\rho$. Hence if $x \in H'$ then $v$ is at distance at most $3\rho/2$ from a center of $C_3$, i.e. $v \in R_3$.

From $x \notin H \cup H'$ we can conclude the existence of $y$ as follows. Consider an optimum center $w \in C^*$ that covers $v$, i.e. $v \in B_w(\rho)$. Recall that $R_1^* \subseteq R_1$ and $R_2^* \subseteq R_2$. Since $v \notin R_1 \cup R_2$, this means that $w$ is neither in $C_1^*$ nor in $C_2^*$ so that $w \in C_3^*$. By definition of $H'$, any hub at distance at most $\rho$ from a center in $C_3^*$ is in $H'$; unless it is in $H$. Hence, as $x \notin H \cup H'$, the distance between $x$ and $w$ must be more than $\rho$. Since $\text{dist}_G(v, x) \leq \rho/2$, we get $\text{dist}_G(v, w) > \rho/2$. We also know that $\text{dist}_G(v, w) \leq \rho$, because $w$ covers $v$. Hence the shortest path cover $\text{spc}(\rho/2)$ must contain the hub $y$, which lies on the shortest path between $v$ and $w$. In particular, $\text{dist}_G(v, y) \leq \rho$ and $\text{dist}_G(y, w) \leq \rho$. Analogous to the argument used for $x$ above, $R_1$ contains all vertices at distance at most $\rho$ from $H$, so that $y \notin H$ since $v \notin R_1$. However, then the distance bound for $y$ and $w$ yields $y \in H'$, as $w \in C_3^*$.

Since $C_1^*$ contains all non-cluster centers but $w \notin C_1^*$, by Lemma 4 we get $\text{dist}_G(y, w) > \rho/2$, which implies $\text{dist}_G(v, y) < \rho/2$. But then $v$ is contained in the ball $B_y(\rho/2)$, which we know is part of the third region $R_3$ since $y \in H'$. This contradicts the assumption that $v$ was not covered by the approximate center set. \[\square\]

Note that the proof of Lemma 7 does not imply that $R_3^* \subseteq R_3$, as was the case for $R_1$ and $R_2$. It suffices though to establish the correctness of the algorithm. The runtime analysis concluding the proof of Theorem 2 is deferred to the full version of the paper.
4 Hardness Results

We begin by observing that the original reduction of Hsu and Nemhauser [19] for $k$-Center also implies that there are no $(2 - \varepsilon)$-FPA algorithms.

**Theorem 8.** It is $W[2]$-hard for parameter $k$ to compute a $(2 - \varepsilon)$-approximation to the $k$-Center problem, for any $\varepsilon > 0$.

We now turn to proving that $(2 - \varepsilon)$-approximations are hard to compute on graphs with low highway dimension. For this we introduce a general reduction from low doubling metrics to low highway dimension graphs in the next lemma. Here the aspect ratio of a metric $(X, \text{dist}_X)$ is the maximum distance between any two vertices of $X$ divided by the minimum distance.

**Lemma 9.** Given any metric $(X, \text{dist}_X)$ with constant doubling dimension $d$ and aspect ratio $\alpha$, for any $\varepsilon > 0$ there is a graph $G = (X, E)$ of highway dimension $O((\log(\alpha)/\varepsilon)^d)$ on the same vertex set such that for all $u, v \in X$, $\text{dist}_X(u, v) \leq \text{dist}_G(u, v) \leq (1 + \varepsilon)\text{dist}_X(u, v)$. Furthermore, $G$ can be computed in polynomial time from the metric.

Feder and Greene [13] show that, for any $\varepsilon > 0$, it is NP-hard to compute a $(2 - \varepsilon)$-approximation for the $k$-Center problem in two-dimensional $L_\infty$ metrics. Furthermore all edges of the instance they construct in the reduction have unit length, and thus the aspect ratio is at most $n$. The doubling dimension of any such metric is 2, since a ball of radius $2r$ (a “square” of side-length $4r$) can be covered by 4 balls of radius $r$ (“squares” of side-length $2r$). By the reduction given in Lemma 9 we thus get the following result.

**Corollary 10.** For any constant $\varepsilon > 0$ it is NP-hard to compute a $(2 - \varepsilon)$-approximation for the $k$-Center problem on graphs of highway dimension $O(\log^2 n)$.

The challenge remains is to push the highway dimension bound of this inapproximability result down to a constant. This would mean that no $(2 - \varepsilon)$-FPA algorithm for $k$-Center exists if the parameter is the highway dimension $h$, unless $P=NP$. However, assuming the exponential time hypothesis, any $(2 - \varepsilon)$-FPA algorithm for parameter $h$ must have doubly exponential running time. In particular, by Corollary 10 any algorithm with running time $2^{2^{O(h)}} \cdot n^{O(1)}$ would solve an NP-hard problem in subexponential $2^{o(n)}$ time when $h \in O(\log^2 n)$. Thus if a $(2 - \varepsilon)$-approximation algorithm for $k$-Center with parameter $h$ exists, it is fair to assume that its running time depending on $h$ must be extremely large.

The following lemma gives further evidence that obtaining a $(2 - \varepsilon)$-FPA algorithm for parameter $h$ is hard. As argued below, it excludes the existence of such algorithms that only use shortest path covers of constant scales.

**Lemma 11.** For any $\varepsilon > 0$ it is NP-hard to compute a $(2 - \varepsilon)$-approximation for the $k$-Center problem on graphs for which on any scale $r > 0$ there is a locally $(3 \cdot 2^{2r-1} + 1)$-sparse shortest path cover $\text{SPC}(r)$. Moreover, this is true for instances where the optimum cost $\rho$ is at most 4.
Consider a \((2-\varepsilon)\)-FPA algorithm for \(k\)-Center, which only takes shortest path covers of constant scales into account, where the parameter is their sparseness. That is, the algorithm computes a \((2-\varepsilon)\)-approximation using hub sets \(\text{spc}(r)\) only for values \(r \leq R\) for some \(R \in O(1)\), and the parameter is a value \(s\) such that for every \(r \leq R\), \(\text{spc}(r)\) is locally \(s\)-sparse. By Lemma 11 such an algorithm would imply that \(P=NP\). Moreover this is true even if \(R \in O(\rho)\). Hence if it is possible to beat the inapproximability barrier of 2 using the local sparseness as a parameter, then such an algorithm would have to take large (non-constant) scales into account. Note that the running time of our \(3/2\)-FPA algorithm can in fact be bounded in terms of the local sparseness of \(\text{spc}(\rho/2)\) instead of the highway dimension. Therefore, by Theorem 8 and Lemma 11, our algorithm necessarily needs to combine the parameter \(h\) with \(k\) in order to achieve its approximation guarantee.

References


A $(1 + \varepsilon)$-EMBEDDING OF LOW HIGHWAY DIMENSION GRAPHS INTO BOUNDED TREewidth GRAPHS

ANDREAS EMIL FELDMANN$^1$, WAI SHING FUNG$^3$, JOCHEN KÖNEMANN$^3$, AND IAN POST$^2$

Abstract. Graphs with bounded highway dimension were introduced by Abraham et al. [Proceedings of SODA 2010, pp. 782–793] as a model of transportation networks. We show that any such graph can be embedded into a distribution over bounded treewidth graphs with arbitrarily small distortion. More concretely, given a weighted graph $G = (V, E)$ of constant highway dimension, we show how to randomly compute a weighted graph $H = (V, E')$ that distorts shortest path distances of $G$ by at most a $1 + \varepsilon$ factor in expectation, and whose treewidth is polylogarithmic in the aspect ratio of $G$. Our probabilistic embedding implies quasi-polynomial time approximation schemes for a number of optimization problems that naturally arise in transportation networks, including Travelling Salesman, Steiner Tree, and Facility Location. To construct our embedding for low highway dimension graphs we extend Talwar’s [Proceedings of STOC 2004, pp. 281–290] embedding of low doubling dimension metrics into bounded treewidth graphs, which generalizes known results for Euclidean metrics. We add several nontrivial ingredients to Talwar’s techniques, and in particular thoroughly analyze the structure of low highway dimension graphs. Thus we demonstrate that the geometric toolkit used for Euclidean metrics extends beyond the class of low doubling metrics.

Key words. metric embeddings, highway dimension, QPTAS, travelling salesman, Steiner tree, facility location

AMS subject classifications. 68W25, 68Q25

DOI. 10.1137/16M1067196

1. Introduction. In [14, 15], Bast and co-authors studied shortest-path computations in road networks and observed that such networks are highly structured: there is a sparse set of transit or access nodes such that when travelling from any point $A$ to a distant location $B$ along a shortest path, one will visit at least one of these nodes. The authors presented a shortest-path algorithm (called transit node routing) that capitalizes on this structure in road networks and demonstrated experimentally that it improves over previously best algorithms by several orders of magnitude. Motivated by the work of Bast and co-authors (among others), Abraham and co-authors [1, 2, 4] introduced a formal model for transportation networks and defined the notion of highway dimension. Informally speaking, an edge-weighted graph $G = (V, E)$ has small highway dimension if, for any scale $r \geq 0$ and for all vertices $v \in V$, shortest paths of length at least $r$ that are close (in terms of $r$) to $v$ are hit by a small set of hub vertices. In the following formal definition, if $\text{dist}(u, v)$ denotes the shortest-path...
distance between vertices $u$ and $v$, let $B_r(v) = \{u \in V | \text{dist}(u,v) \leq r\}$ be the ball of radius $r$ centred at $v$. We will also say that a path $P$ lies inside $B_r(v)$ if all its vertices lie inside $B_r(v)$.

**Definition 1.1.** The highway dimension of a graph $G$ is the smallest integer $k$ such that, for some universal constant $c \geq 4$, for every $r \in \mathbb{R}^+$, and every ball $B_{cr}(v)$ of radius $cr$, there are at most $k$ vertices in $B_{cr}(v)$ hitting all shortest paths of length more than $r$ that lie in $B_{cr}(v)$.

Rather than working with the above definition directly, we often consider the closely related notion of shortest path covers (also introduced in [1]).

**Definition 1.2.** For a graph $G$ and $r \in \mathbb{R}^+$, a shortest path cover $\text{spc}(r) \subseteq V$ is a set of hubs that intersect all shortest paths of length in $(r, cr/2]$ of $G$. Such a cover is called locally $s$-sparse for scale $r$ if no ball of radius $cr/2$ contains more than $s$ vertices from $\text{spc}(r)$.

In particular, a graph with highway dimension $k$ can be seen to have a locally $k$-sparse shortest path cover for any scale $r$ [1] (using the same constant $c$ in Definitions 1.1 and 1.2). In both definitions above, Abraham et al. [1] specifically chose $c = 4$ but also noted that this choice is, to some extent, arbitrary. In the present paper, the flexibility of being able to choose a slightly larger value of $c$ is crucial as we will explain shortly. In the following, we will let $\lambda = c - 4$ and call it the violation of Abraham et al.’s original definition. While we believe that a small positive violation does not stray from the intended meaning of highway dimension, we also point out that there are graphs whose highway dimension is highly sensitive to the value of $c$, as we explain in section 9. Hence this is not an entirely innocuous change.

Abraham and co-authors [1, 2, 4] focused on the shortest-path problem and formally investigated the performance of various prominent heuristics as a function of the highway dimension of the input graph. They also pointed out that “conceivably, better algorithms for other optimization problems can be developed and analysed under the small highway dimension assumption.” This statement is the starting point of this paper.

We study three prominent NP-hard optimization problems that arise naturally in transportation networks: Travelling Salesman, Steiner Tree and Facility Location. (See section 8 for formal definitions.) Each of these was first studied in the context of transportation networks, and as we will show they admit quasi-polynomial time approximation schemes (QPTASs) on graphs with bounded highway dimension. Our work thereby provides a complexity-theoretic separation between the class of low highway dimension graphs and general graphs, in which the aforementioned problems are APX-hard [23, 25, 29].

Technically, we achieve the above results by employing the powerful machinery of metric space embeddings [12, 26]. Specifically, for any $\varepsilon > 0$ we probabilistically compute a low-treewidth graph $H$ on the same vertex set as the input graph $G$ such that the shortest-path distance between any two vertices in $H$ is lower bounded by their distance in $G$, and, in expectation, upper bounded by $1 + \varepsilon$ times their distance in $G$. The latter factor by which the distances are bounded from above is called the distortion or stretch of the embedding $H$. (See section 2 for formal definitions.) The following is the main result of this paper, where the aspect ratio is the maximum distance of any two vertices divided by the minimum distance between any vertices.

**Theorem 1.3.** Let $G$ be a graph with highway dimension $k$ of violation $\lambda > 0$ and aspect ratio $\alpha$. For any $\varepsilon > 0$, there is a polynomial-time computable probabilistic embedding $H$ of $G$ with treewidth $(\log \alpha)^{O((\log^2(\frac{1}{\varepsilon})/\lambda)}$ and expected distortion $1 + \varepsilon$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Low highway dimension graphs do not exclude fixed-size minors and therefore do not have low treewidth [35]: the complete graph on vertices \( \{1, \ldots, n\} \), where each edge \( \{i, j\} \) with \( i < j \) has length \( c' \), has highway dimension 1. The example also shows that the aspect ratio of a low-highway dimension graph can be exponential. Using standard techniques, we will show that the aspect ratio may be assumed to be polynomial for our considered problems when aiming for \( 1 + \varepsilon \) approximations. Existing algorithms for bounded treewidth graphs [6, 16] then imply QPTASs on graphs with constant highway dimension. (See section 8 for more details.)

While Travelling Salesman, Facility Location, and Steiner Tree are APX-hard in general graphs, improved algorithms are known in special cases. For example, polynomial time approximation schemes (PTASs) for all three of these problems are known if the input metric is low-dimensional Euclidean or planar [5, 7, 9, 16, 18, 31, 34]. Talwar [38] also showed that the work in [7, 9, 34] extends (albeit with quasi-polynomial running time) to low doubling dimension metrics. Bartal, Gottlieb, and Krauthgamer [13] later presented a PTAS for Travelling Salesman instances in this class.

The concept of doubling dimension was studied by Gupta, Krauthgamer, and Lee [30] and captures metrics that have bounded growth. Formally, a metric space \((X, \text{dist})\) has doubling dimension \(d\) if \(d\) is the smallest number such that every ball of radius \(2r\) is contained in the union of balls \(2^d\) balls of radius \(r\). The class of constant doubling dimension metrics strictly generalizes that of Euclidean metrics in constant dimensions. Doubling dimension and highway dimension (as defined here) are incomparable metric parameters; however, Abraham et al. [1] noted that grids have doubling dimension 2 but highway dimension \(\Theta(\sqrt{n})\), while stars have doubling dimension \(\Theta(\log n)\) and highway dimension 1.

We briefly note here that there are alternative definitions of highway dimension. (See section 9 for a detailed discussion.) In particular, the more restrictive definition in [4] implies low doubling-dimension, and hence Talwar [38] readily yields a QPTAS for the optimization problems we study. Our choice of definition is deliberate, however, and motivated by the fact that Definition 1.1 captures natural transportation networks that the more restrictive definition does not. For instance, typical hub-and-spoke networks used in air traffic models are nonplanar and have high doubling dimension, since they feature high-degree stars. This immediately renders them incompatible with the highway dimension definition in [4]. Nevertheless they have low highway dimension by Definition 1.1, since the airports act as hubs, which become sparser with growing scales as longer routes tend to be serviced by bigger airports. We also prove in section 9 that our definition is a strict generalization of the one in [4]: any graph with highway dimension \(k\) according to [4] has highway dimension \(O(k^2)\) according to Definition 1.1, while a corresponding lower bound is not possible in general.

Our results not only provide further evidence that the highway dimension parameter is useful in characterizing the complexity of graph theoretic problems in combinatorial optimization, but, importantly, they also show that the geometric toolkit of [7, 9, 34] extends beyond the class of low doubling dimension metrics, since the proof of Theorem 1.3 heavily relies on the embedding techniques proposed in [38].

1.1. Our techniques. The embedding constructed in the proof of Theorem 1.3 heavily relies on previous work by Talwar [38] but needs many nontrivial new ideas, a few of which we sketch here.

Talwar’s embedding algorithm first computes a so-called split-tree decomposition, a certain laminar family of subsets of the set \(X\) of points underlying the given metric.
space. Initially, this family contains just one element, the set $X$ itself. In each step, the algorithm picks a nonsingleton leaf $C$ of the family, partitions it into sets $C_1, \ldots, C_q$ of random diameter roughly half of that of $C$, and adds these to the family. The algorithm continues until all the leaves in the family are singletons. An element $C$ of the computed decomposition is commonly referred to as a cluster.

Each cluster $C$ of the split-tree decomposition is associated with a set of net points; net points are well spaced in $C$, and each point in $C$ is close to at least one of these. For each cluster, only the edges between the net points of its child clusters are kept to form the embedding. The shortest path between two points can then be approximated by a path that exits each cluster only via the net points. The error introduced due to the shifting of points on a path to net points, as well as the total distortion, can be bounded as the sum of errors over all levels of the split-tree decomposition. In the tree decomposition (see section 2 for formal definitions) of the resulting embedding, each bag corresponds to a cluster and consists of the net points of its child clusters. Using the bounded doubling dimension assumption, the number of child clusters and number of net points per cluster can be bounded by constants depending on the doubling dimension and the desired stretch. This in turn bounds the embedding’s treewidth.

We want to construct a similar recursive decomposition for metrics with low high-way dimension, but this turns out to be nontrivial. In order to obtain a decomposition we observe that the hubs in the shortest path cover induce a natural clustering of the vertices in $G$ for any scale $r$ (see Figure 1). Each vertex $v \in V$ whose distance from any hub is larger than $2r$ is said to belong to a town that is contained in the ball of radius $r$ centered at $v$. All vertices that are not part of a town (and hence at distance no more than $2r$ from some hub) are said to be part of the sprawl. We will show that towns are nicely separated from other towns and the sprawl and that the degree of separation is highly sensitive to the choice of $c$ in Definition 1.1. It turns out that choosing $c = 4$ yields a separation that is just barely too small.

Based on this clustering, we compute a hierarchical decomposition of the graph that we call the towns decomposition. It is a laminar family of towns and recursively separates the graph into towns of decreasing scales, and our embedding is computed recursively on this decomposition. The towns decomposition is analogous to the quad-tree decomposition in PTASs for Euclidean metrics [7, 8, 9, 10] or the split-tree decomposition for low doubling dimension metrics [38], though the particulars differ greatly. At a high level, towns look similar to clusters in Talwar’s split-tree decomposition. However, while in Talwar’s split-tree decomposition, clusters have a relatively small number of child clusters, towns can contain a very large number of child towns. As it turns out, however, these child towns are connected through hubs of higher scales, which can be chosen in a way such that they have bounded doubling dimension. We can therefore apply Talwar’s decomposition technique to
these connecting hubs. We then recursively construct a low treewidth embedding for each child town and attach these embeddings to the embedding of the connecting hubs. The details are described in section 4.

The most intricate part of our result is to prove a low doubling dimension of these “connecting hubs,” which are chosen as follows. We prove that to preserve all distances within a town $T$ it suffices to connect embeddings of $T$’s child towns in the towns decomposition via a carefully chosen set of so-called core hubs within $T$. To prove low doubling dimension, the general idea is to rely on the local sparsity of the shortest path covers (see Figure 2): by definition, the core hubs lie in the sprawls of various scales, and for scale $r$ the sprawl can be covered by balls of radius $2r$ around the hubs of the shortest path cover. In a low highway dimension graph, any ball $B$ of radius $cr/2$ contains only a small number of hubs. Hence, to bound the doubling dimension, we attempt to use these hubs as centers of balls of smaller radius to cover the core hubs. These balls have radius $2r < cr/2$, and hence this scheme can be applied recursively in order to cover the core hubs in $B$ with balls of half the radius. Several issues arise with this approach though. For instance, part of the sprawl for scale $r$ in $B$ might be covered by balls centered at hubs outside of $B$. However a key insight of our work is that in fact the number of hubs in the vicinity of a ball is also bounded when using Definition 1.1 for the highway dimension (see Lemma 6.2).

Another obstacle when trying to bound the doubling dimension of the core hubs is that, unlike the nets in Talwar’s split-tree decomposition, the hubs do not form a hierarchy, i.e., a hub at some scale may not be a hub at a lower scale. Nevertheless, we show that core hubs at different scales can be aligned: they can be shifted slightly in order to obtain a nested structure. We are able to show that this alignment process does not affect the target stretch of our embedding and, most importantly, ensures that the resulting set of approximate core hubs within $T$ has a small doubling dimension. We may thus apply Talwar’s [38] embedding of low doubling dimension metrics into bounded treewidth graphs to the approximate core hubs.

1.2. Related work. The highway dimension concept was introduced by Abraham et al. [1], who showed that the efficiency of certain shortest-path heuristics can be explained with this parameter. Follow-up papers [2, 4] introduced alternative definitions and showed that it is possible to approximate the highway dimension $k$ within an $O(\log k)$ factor assuming that shortest paths are unique. For the $p$-Center problem the embedding techniques given in this paper are not applicable since the objective function is nonlinear. Instead, in [27] a parameterized approximation for this problem on low highway dimension graphs is presented. Bauer et al. [17] show that for any graph $G$ there exist edge lengths such that the highway dimension is $\Omega(pw(G) / \log n)$.
where \(pw(G)\) is the pathwidth of \(G\). Also Kosowski and Viennot [32] consider the highway dimension and compare it to the related skeleton dimension.

In the seminal work of Bartal [11, 12] it was shown that any graph can be embedded into a distribution over trees with an expected polylogarithmic stretch. The stretch bound was later improved to \(O(\log n)\) by Fakcharoenphol, Rao, and Talwar [26], which is the best possible. These techniques led to the embedding of low doubling dimension metrics into bounded treewidth graphs by Talwar [38], which forms a major ingredient in our result. Another generalization is that of Chan and Gupta [22], who showed how to embed a metric of low correlation dimension into a metric of bounded treewidth. It is worth noting that the highway dimension cannot be bounded in terms of the correlation dimension (due to the complete graph example described above). In terms of lower bounds, there are graphs [20, 21] with treewidth \(t\), which cannot be embedded into distributions over graphs excluding minors of size \(t - 1\), without incurring an expected stretch of \(\Omega(\log n)\). The authors also show that embeddings of planar graphs into bounded treewidth graphs must incur logarithmic distortions.

2. Embeddings for low doubling dimension metrics. Next we formally define the treewidth and summarize the properties of Talwar’s [38] embedding for low doubling dimension metrics that we require for our construction. More details will be given in section 5, which are needed for the analysis of the stretch of our embedding.

Let \(G = (V,E)\) be a graph. For \(u,v\in V\) we denote the length of the shortest path between \(u\) and \(v\) by \(\text{dist}(u,v)\) and the distance between two sets \(S,T\subseteq V\) by \(\text{dist}(S,T) = \min_{u\in S,v\in T} \text{dist}(u,v)\). If the metric used for distances is ambiguous we specify the graph in the subscript, such as \(\text{dist}_G(u,v)\) or \(\text{dist}_H(u,v)\). The diameter \(\text{diam}(\cdot)\) of a graph or set of vertices is the maximum distance between any two vertices. The treewidth of a graph measures how close the graph is from being a tree. A tree decomposition of a graph \(G\) consists of a tree whose vertices are labelled by subsets of \(V\) that are commonly referred to as bags. We will often identify the bags with the vertices of the tree and talk about a “tree of bags.” Bags satisfy certain structural properties, as is formalized in the following definition.

Definition 2.1. A tree decomposition \(D\) of a graph \(G = (V,E)\) is a tree \(T\) each of whose vertices \(v\) are labelled by a bag \(b_v \subseteq V\) of vertices of \(G\). We require the following properties:

(a) \(\bigcup_{v \in V(T)} b_v = V\),

(b) for every edge \((u,w)\in E\) there is a vertex \(v \in V(T)\) such that \(b_v\) contains both \(u\) and \(w\), and

(c) for every \(v \in V\) the set \(\{u \in V(T) : v \in b_u\}\) induces a connected subtree of \(T\).

The width of the tree decomposition is \(\max\{|b_v| - 1 : v \in V(T)\}\). The treewidth of a graph \(G\) is the minimum width among all tree decompositions for \(G\).

To construct our embedding we will mainly focus on the shortest path metric of the graph \(G\). We let the distance function of every considered metric be the function \(\text{dist}(\cdot,\cdot)\) of the underlying graph. Though the treewidth is a property of a graph’s edge set, whereas doubling dimension is a property of the metric it defines, Talwar [38] shows that low doubling dimension graphs can be approximated to within \(1 + \varepsilon\) by bounded treewidth graphs. Formally this means the following.

Definition 2.2. Let \((X, \text{dist})\) be a metric and \(\mathcal{D}\) be a distribution over metrics \((X, \text{dist}^\prime)\). If for all \(x,y \in X\), \(\text{dist}(x,y) \leq \text{dist}^\prime(x,y)\) for each \(\text{dist}^\prime \in \mathcal{D}\), and \(E_{\text{dist}^\prime \in \mathcal{D}}[\text{dist}^\prime(x,y)] \leq a \cdot \text{dist}(x,y)\), then \(\mathcal{D}\) is an embedding with (expected) stretch or
EMBEDDING HIGHWAY DIMENSION INTO BOUNDED TREewidth

If every \( \delta \) is the shortest path metric of some graph class \( G \), then \( D \) is a \((\text{probabilistic})\) embedding into \( G \).

The main result of Talwar [38] that we use for our embedding of low highway dimension graphs into bounded treewidth graphs is the following.

**Theorem 2.3** (see [38]). Let \((X, \text{dist})\) be a metric with doubling dimension \(d\) and aspect ratio \(\alpha\). For any \(\varepsilon > 0\), there is a polynomial-time computable probabilistic embedding \(H\) of \((X, \text{dist})\) with treewidth \((d \log(\alpha)/\varepsilon)^{O(d)}\) and expected distortion \(1 + \varepsilon\).

As described in the introduction, Talwar’s embedding employs a randomized split-tree decomposition, which is a hierarchical decomposition of the vertices \(X\) of a metric into clusters of smaller and smaller diameter. A cluster is a subset of \(X\), which is partitioned into clusters at most half the diameter on the next lower level, so that the highest cluster is \(X\) itself and the lowest ones are individual vertices. The geometrically decreasing diameters of the levels are set according to a random variable. Each level of this hierarchy is associated with an index. Our construction of the embedding for low highway dimension graphs also has levels associated with indices, but these have different growth rates. To avoid confusion we will denote the levels of Talwar’s split-tree decomposition with indices \(i, j\), etc., and ours with indices \(i, j\), etc.

The tree decomposition constructed from the split-tree has a bag for each cluster. The tree on the bags exactly corresponds to the split-tree. Each bag contains a coarse set of points of the cluster. More concretely it contains a \(\delta\)-standard, defined as follows.

**Definition 2.4.** For a metric \((X, \text{dist})\), a subset \(Y \subseteq X\) is called a \(\delta\)-cover if for every \(u \in X\) there is a \(v \in Y\) such that \(\text{dist}(u, v) \leq \delta\). A \(\delta\)-net is a \(\delta\)-cover with the additional property that \(\text{dist}(u, v) > \delta\) for all vertices \(u, v \in Y\).

For a cluster \(C\) on level \(i\) the corresponding bag contains a \(\Theta(\varepsilon 2^i/(d \log \alpha))\)-net of \(C\). For every bag \(b\) the graph embedding contains a complete graph on the nodes in \(b\) with edge lengths corresponding to distances in the metric. The net in each bag serves as a set of portals, through which connections leaving the cluster are routed, analogous to those in [8].

3. Properties of low highway dimension graphs. We assume without loss of generality (w.l.o.g.) that every shortest path in our input graph is unique by slightly perturbing edge lengths. This allows us to compute locally \(O(k \log k)\)-sparse shortest path covers in polynomial time [2] (or locally \(k\)-sparse covers in time \(n^{O(k)}\)). We show in section 9 that computing the highway dimension is \(\text{NP}\)-hard even for graphs with unit edge lengths, so in general approximations are needed.

An important observation is that the vertices of low highway dimension graphs are grouped together in all regions that are far from the hubs. This gives rise to our main observation on the structure of low highway dimension graphs, as summarized in the following definition: for any scale the vertices are partitioned into one sprawl and several towns with large separations in between.

**Definition 3.1.** Given a shortest path cover \(\text{spc}(r)\) for scale \(r\), for any vertex \(v \in V\) such that \(\text{dist}(v, \text{spc}(r)) > 2r\), we call the set \(T = \{u \in V|\text{dist}(u, v) \leq r\}\) a town for scale \(r\). The sprawl for scale \(r\) is the set of all vertices that are not in towns.

Note that the vertices of the sprawl are at most \(2r\) away from a hub, but there can be vertices in towns that are closer than \(2r\) to some hub, as long as the town has some other vertex that is further away. Note also that the towns are defined with respect to a shortest path cover \(\text{spc}(r)\), and using two distinct shortest path covers will not always result in the same set of towns. We will fix an inclusionwise minimal
shortest path cover \( \text{spc}(r) \) for any scale \( r \) and only consider towns with respect to this cover. We summarize the basic properties of towns below.

**Lemma 3.2.** Let \( T \) be a town of scale \( r \). Then \( \text{diam}(T) \leq r \) and \( \text{dist}(T, V \setminus T) > r \). For any vertex \( v \) of the sprawl of scale \( r \), \( \text{dist}(v, \text{spc}(r)) \leq 2r \).

**Proof.** The bound on the distance from any vertex of the sprawl to the nearest hub follows immediately from the definition of the towns. To prove that the diameter of a town \( T \) is at most \( r \), assume there are vertices \( u, w \in T \) such that \( \text{dist}(u, w) > r \). By Definition 3.1 we know there is a vertex \( v \in T \) such that \( \text{dist}(u, v) \leq r \) and \( \text{dist}(w, v) \leq r \), so that \( \text{dist}(u, v) \leq 2r \). This means that the length of the shortest path between \( u \) and \( w \) lies in the interval \((r, cr/2] \), as by Definition 1.1 the constant \( c \) defining \( \text{spc}(r) \) is at least 4. In particular, there is a hub \( h \in \text{spc}(r) \) that lies on this shortest path. Assume w.l.o.g. that \( h \) is closer to \( w \) than to \( u \), so that \( \text{dist}(h, w) \leq r \). But then, \( \text{dist}(h, v) \leq \text{dist}(h, w) + \text{dist}(w, v) \leq 2r \), which contradicts \( \text{dist}(v, \text{spc}(r)) > 2r \).

Similarly, we can prove that the distance of any vertex \( u \) of a town \( T \) to any vertex \( w \) outside of \( T \) is more than \( r \). Consider again the vertex \( v \in T \) given by Definition 3.1, for which \( \text{dist}(u, v) \leq r \), \( \text{dist}(w, v) > r \), and \( \text{dist}(v, \text{spc}(r)) > 2r \). If we assume that \( \text{dist}(w, u) \leq r \), then from the first distance bound for \( u \) and \( v \) we get \( \text{dist}(w, v) \leq 2r \). Together with \( \text{dist}(w, v) > r \), this means that the length of the shortest path \( P \) between \( w \) and \( v \) lies in the interval \((r, cr/2] \), as by Definition 1.1 \( c \geq 4 \). Hence there is a hub \( h \in \text{spc}(r) \) on \( P \) that is at most as far from \( v \) as \( w \) is, i.e., \( \text{dist}(v, h) \leq 2r \). However this contradicts \( \text{dist}(v, \text{spc}(r)) > 2r \).

We will exploit the structure given by Lemma 3.2 for growing scales to construct our embedding. More concretely, we will consider scales \( r_i = (c/4)^i \) for values \( i \in \mathbb{N}_0 \) and call \( i \) the level of the sprawl, towns, and shortest path cover of scale \( r_i \). We choose our scales in this way since \( 2r_i = cr_{i-1}/2 \). As a consequence, a ball of radius \( 2r_i \) around a hub of level \( i \) that covers part of the sprawl contains at most \( s \) hubs of the next lower level \( i - 1 \) if the shortest path covers are locally \( s \)-sparse. We will exploit this in our analysis in order to bound the treewidth of our embedding.

Note that the scales are monotonically nonincreasing if we choose \( c \leq 4 \). As we shall see, positive scale-growth is essential, however, for our algorithm as it allows us to argue that any two disjoint towns are sufficiently separated.

Throughout this paper we will assume that the shortest path covers are inclusion-wise minimal. By scaling we can assume that the shortest distance between any two vertices is slightly more than \( c/2 \). Hence \( \text{spc}(r_0) = \emptyset \) since there are no paths of length in \((r_0, cr_0/2] \). In particular this means that on level 0 there is no sprawl, and each vertex forms a singleton town. The highest level we consider is \( m = \lceil \log_{c/4} \text{diam}(G) \rceil \). At this level \( \text{spc}(r_m) = \emptyset \) and hence the whole vertex set \( V \) of the graph is a town.

We show next that towns of different levels form a laminar family \( \mathcal{T} \). Due to this laminar structure of towns we will use tree terminology such as parents, children, siblings, ancestors, and descendants of towns in \( \mathcal{T} \). Note that these family relations are with respect to the laminarity of \( \mathcal{T} \) and not the levels on which towns exist. The root of the laminar family is the highest level town \( V \).

**Lemma 3.3.** Given a graph \( G \), the set \( \mathcal{T} := \{ T \subseteq V \mid T \text{ is a town on level } i \in \mathbb{N}_0 \} \) forms a laminar family. Furthermore, any town \( T \in \mathcal{T} \) on level \( i \) either has 0 or at least 2 child towns, and in the latter case these are towns on levels below \( i \).

**Proof.** Assume \( \mathcal{T} \) is not laminar, in which case there are two towns \( T_1 \) and \( T_2 \) in \( \mathcal{T} \) that cross, i.e., all of the sets \( T_1 \cap T_2 \), \( T_1 \setminus T_2 \), and \( T_2 \setminus T_1 \) are nonempty. Assume that \( T_1 \) is a town of level \( i \), while \( T_2 \) is a town of level \( j \geq i \). Let \( v \) and \( w \) be two
vertices of $T_1$ such that $v \in T_2$ but $w \notin T_2$. By Lemma 3.2, $\text{dist}(v, w) \leq \text{diam}(T_1) \leq r_i$ and $\text{dist}(v, w) \geq \text{dist}(T_2, V \setminus T_2) > r_j \geq r_i$—a contradiction.

For the second part, let $T$ be a town in the set $\mathcal{T}$ with a child $T'$. Note that $T \setminus T' \neq \emptyset$, while every vertex is a town on level 0. So there must be another town that is a child of $T$. Now assume there is a town $T$ on level $i$ with a child town $T'$ on level $j \geq i$. By Lemma 3.2, the diameter of $T$ is at most $r_i$, and any other child town of $T$ must be at distance more than $r_j \geq r_i$, from $T'$. This would mean that $T$ only has one child town—a contradiction.\qed

The above lemma proves that the following procedure has a well-defined output: starting with a town $T$ on some level $i$, repeatedly remove child towns on level $i - 1$ until only the sprawl remains. Continue by removing all towns on level $i - 2, i - 3$, etc., from the remaining nodes until all nodes have been removed. Then recurse on each of the removed child towns.

Starting the decomposition with town $G$ on level $\log_{\sqrt{4}} \text{diam}(G)$, we refer to the resulting laminar family $\mathcal{T}$ as the towns decomposition of $G$. Note that $\mathcal{T}$ partitions every town $T \in \mathcal{T}$, and although $T$ appears once in $\mathcal{T}, T$ can be a town on multiple levels of the shortest path covers, if it is a town with respect to both $\text{spc}(r_i)$ and $\text{spc}(r_{i+1})$. From now on we will consider the graph metric $(V, \text{dist}_G)$ induced by $G$ instead of $G$ itself. All properties of towns and sprawl, such as given by Lemmas 3.2 and 3.3, are still valid in the metric.

4. Constructing the embedding. We now describe our algorithm in more detail. PTASs for Euclidean and low doubling metrics [8, 38] use hierarchical decompositions coupled with a small number of “portal” nodes: paths leaving a cluster in the decomposition must do so via an appropriate portal, resulting in a small “interface” between distinct clusters in the decomposition. Intuitively, the hubs are natural choices for portals, since long paths through some ball must pass through a hub. However problems crop up almost immediately because hubs are not guaranteed to be well-spaced or consistent between levels, and although all long paths through a ball may be hit by portals, there may be many short paths that go nowhere near one.

We overcome these difficulties by exploiting the properties of the towns decomposition. Lemma 3.2 guarantees that towns are isolated from both each other and the sprawl. Consequently, any approximate shortest path between nodes in a town must remain within that town. The embedding is constructed recursively on the metric using the structure of the towns decomposition $\mathcal{T}$. That is, for a town $T \in \mathcal{T}$ we assume that we have already computed an embedding (and accompanying tree decomposition) with expected stretch $1 + \varepsilon$ for each child town of $T$. We then connect these embeddings so that distances between them are preserved within a $1 + \varepsilon$ factor in expectation. This gives an embedding for $T$ and, since $V$ itself is the root of the towns decomposition, eventually yields an embedding for $G$.

The key insight that lets us connect the child towns of $T$ is that there exists a set of so-called approximate core hubs $X_T$ in $T$ with low doubling dimension that can serve as the crossroads through which child towns connect. We will compute a low-treewidth embedding of the set $X_T$ based on Theorem 2.3 and connect the embeddings of the child towns to it. In particular, for every child town $T'$ we will identify a bag $b$ of the tree decomposition of $X_T$ containing hubs that are close to $T'$. We call $b$ the connecting bag of $T'$. The embedding of $T$ is constructed by connecting every vertex in each child town to every hub in the corresponding connecting bag. As we show in section 5, this means that short connections between child towns can be routed directly through hubs in the connecting bags. Long connections on the
Fig. 3. The cores of three different levels of town $T$ (enclosed by dotted lines for levels $i < j$). Note that some hubs of level $j-2$ (small crosses) lie in towns of level $j-1$ (larger dashed circles), and these are not core hubs.

other hand can be routed through the embedding of the core hubs $X_T$ at only a small overhead.

The tree decomposition for $T$ is constructed by connecting each tree decomposition $D_T$ for a child town $T'$ to the corresponding connecting bag $b$ of the tree decomposition $D_X$ for the hubs in $X_T$ (lines 29 to 31 in Algorithm 1). Even though this yields a tree of bags containing all vertices of the town $T$, properties (b) and (c) of Definition 2.1 might be violated by this initial attempt. As we will show in section 7, we need to make two modifications to the bags: first we need to add all vertices of $b$ to each bag of $D_T'$. Since the treewidth of $D_X$ is bounded by Theorem 2.3, this does not increase the sizes of bags by much. Second, we also need to add all hubs of $X_T$ within the child town $T'$ to each bag of $D_T'$, as well as to $b$ and all descendants of $b$ in $D_X$. To bound the growth of the bags in this step, we need to bound the number of hubs in $X_T$ in a child town $T'$, which we do in section 7.

The set $X_T$ is an approximate hub set of $T$. To define the set properly we need some additional insights on the structure of hubs of different levels in $T$. The core of $T$ is the intersection of sprawls formed by removing all child towns of $T$ above a given level (c.f. Figure 3).

**Definition 4.1.** Let $T \in \mathcal{T}$ be a town on level $j$, and let $S_i$ be the sprawl of $V$ on level $i \leq j$. The core $C_i$ of $T$ on level $i$ is inductively defined as follows: $C_j = T$, and $C_i = S_i \cap C_{i+1}$ for $i \leq j-1$. The core hubs of $T$ are given by the set $\bigcup_{i=0}^{j-1} C_i \cap \text{spc}(r_i)$.

By this definition a town $T$ on level $j$ can be partitioned into its core on level $i$ and its child towns on levels $\{i, \ldots, j-1\}$. Observe also that the set system $\{C_i\}_{i=0}^{j-1}$ given by the cores forms a chain, i.e., $C_i \subseteq C_i$. Intuitively, the core hubs should have low doubling dimension: if the shortest path covers are locally $s$-sparse, then in a ball around a hub at level $i$ there will be at most $s$ hubs in that ball on level $i-1$, and the balls of half the radius around these hubs cover the core on that level (c.f. Figure 2). In fact one can show that the doubling dimension of the core hubs is fairly small but unfortunately not small enough for our purposes. In particular, we need the doubling dimension to be independent of the aspect ratio $\alpha$ of the metric. To circumvent this issue, roughly speaking, we shift each core hub so that it overlaps with lower level core hubs if possible, making the hubs nested to some degree. However, in order to preserve distances we will only shift them by at most an $\varepsilon$ fraction. This shifting produces the set $X_T$ of approximate core hubs of $T$, which we use to construct our core embedding. Note that we do not use the approximate core hubs $X_T$ to define our towns decomposition, only to produce a low-treewidth core embedding. (See lines 7
EMBEDDING HIGHWAY DIMENSION INTO BOUNDED TREewidth

1677

and 3 in Algorithm 1.) We rely on the following nontrivial properties, which require an intricate proof provided in section 6.

Theorem 4.2. Let \( T \) be a towns decomposition of a graph of highway dimension \( k \), given by locally s-sparse shortest path covers on all levels with violation \( \lambda > 0 \). For any town \( T \in \mathcal{T} \) of a level \( j \) there exists a polynomially computable set of approximate core hubs \( X_T \subseteq T \) such that

- for any core hub \( h \in C_i \cap \text{spc}(r_i) \) of \( T \) on level \( i \in \{1, \ldots, j-1\} \), there is a vertex \( h' \in X_T \) with \( \text{dist}_G(h, h') \leq \varepsilon r_i \), and
- the doubling dimension of \( X_T \) is \( d = O(\log(\frac{h_k \log(1/\varepsilon)}{\lambda})/\lambda) \).

From now on, we use \( d \) to denote the above doubling dimension bound for \( X_T \). Our algorithm computes the low-treewidth embedding \( H_T \) of \( T \) by explicitly computing its tree-decomposition \( D_T \). The latter is constructed by connecting the recursively computed tree decompositions \( D_T \) for child towns \( T' \) of \( T \) to the tree decomposition \( D_X \) of \( H_X \) for the metric induced by the approximate core hubs \( X_T \). For this to work we need to make sure that the approximate core hubs contained in the same child town \( T' \) do not end up in different bags in the tree decomposition \( D_T \) of \( H_T \). Our solution is to pick a representative core hub for each child town \( T' \). Specifically, let \( Y_T \subseteq X_T \) contain one arbitrary approximate core hub for each child town \( T' \) for which \( T' \cap X_T \neq \emptyset \). We say that a vertex \( v \in Y_T \) of a child town \( T' \) represents the nodes in \( X_T \cap T' \) (including \( v \) itself). The submetric \( Y_T \) of \( X_T \) inherits the doubling dimension bound of Theorem 4.2, since the doubling dimension of any submetric is at most twice the doubling dimension of the original metric. This was already noted in [30], and we give a formal proof of this fact in the following. We state this observation slightly more generally than we need it here, as we will reuse it in section 6: in the next lemma the metric \( Z \) is not required to have bounded doubling dimension, but the premise is clearly fulfilled if it does.

Lemma 4.3. Let \((Z, \text{dist})\) be a metric and \(Z' \subseteq Z\). If for every ball \( B_{2r}(v) \subseteq Z \) of radius \( 2r \) there are at most \( 2^d \) balls \( B_r(u_i) \subseteq Z \), with centers \( u_i \) and each with radius \( r \), such that their union contains all vertices in \( B_{2r}(v) \cap Z' \), then the doubling dimension of \((Z', \text{dist})\) is at most \( 2d \).

Proof. Any ball in \((Z', \text{dist})\) corresponds to a ball in \((Z, \text{dist})\) with a center vertex in \(Z'\). Pick a ball \( B_{2r}(v) \subseteq Z \) with radius \( 2r \) and \( v \in Z' \). For each of the \( 2^d \) balls \( B_r(u_i) \) that exist for \( B_{2r}(v) \), there again are at most \( 2^d \) balls \( B_{r/2}(w_{ij}) \) that exist for \( B_{r/2}(w_{ij}) \). Pick any vertex \( w_{ij} \in Z' \) (if any) in such a ball \( B_{r/2}(w_{ij}) \) and consider the ball \( B_r(w_{ij}) \) of double the radius. This ball must contain \( B_{r/2}(w_{ij}) \). Doing this for all such balls \( B_{r/2}(w_{ij}) \) gives at most \( 2^{2d} \) balls, each with a center vertex in \(Z'\), such that their union covers \( B_{2r}(v) \cap Z' \). Hence the ball \( B_{2r}(v) \cap Z' \) in \((Z', \text{dist})\) is covered by at most \( 2^{2d} \) balls in \((Z', \text{dist})\) by intersecting each of these balls in \((Z, \text{dist})\) with \(Z'\).

By Lemma 4.3 the doubling dimension of \( Y_T \) is at most \( 2d \), and so we can compute an embedding \( H_T \) for the metric \((Y_T, \text{dist}_Y)\) with bounded treewidth by Theorem 2.3. Given \( H_T \), together with a tree decomposition \( D_T \) we convert it into an embedding \( H_X \) of \( X_T \) together with a tree decomposition \( D_X \) by replacing a vertex \( v \in Y_T \) with all approximate core hubs that \( v \) represents. (See lines 13 to 15 in Algorithm 1.) In particular, the tree decomposition \( D_X \) of \( H_X \) is obtained from the decomposition \( D_Y \) of \( H_Y \) by replacing \( v \in Y_T \) with all the hubs it represents in each bag containing \( v \). For every bag \( b \) of \( D_X \) the embedding \( H_X \) contains a complete graph on the vertices of \( b \), where the length of an edge \( \{u, v\} \) is the distance \( \text{dist}_G(u, v) \) in \( G \). It is easy to see...
Algorithm 1: Compute embedding $H$ with tree decomposition $D_H$ of graph $G$

for $i = 0, \ldots, \lfloor \log_{\epsilon/4} \text{diam}(G) \rfloor$ do

$\text{spc}(r_i) \leftarrow$ locally $O(k \log k)$-sparse minimal shortest path cover
// See [2]

$T \leftarrow$ towns decomposition based on $\text{spc}(r_i)$

$(H, D_H) = \text{Embed}(V, \lfloor \log_{\epsilon/4} \text{diam}(G) \rfloor)$ // Recursively compute
embedding $H$ with tree decomposition $D_H$

function $\text{Embed}(T, j)$ // Low-treewidth embedding of town $T$ at
level $j$

if $j = 0$ then return $(T, T)$ // A town is a singleton at level 0

Compute approximate core hubs $X_T$ of $T$ // According to
Theorem 4.2

$\text{Towns} \leftarrow \emptyset$ // Set of embeddings of child towns of $T$

for $i = j - 1, \ldots, 0$ do // Recurse on child towns

foreach child town $T' \in T$ of $T$ on level $i$ do

$(H_{T'}, D_{T'}) \leftarrow \text{Embed}(T', i)$

Add $(H_{T'}, D_{T'}, i)$ to $\text{Towns}$

// Compute embedding $H_X$ for $X_T$ with tree decomposition $D_X$

$Y_T \leftarrow$ one node in $X_T \cap T'$ for each child town $T'$ of $T$ for which

$X_T \cap T' \neq \emptyset$

$(H_Y, D_Y) \leftarrow \text{Talwar}(Y_T, \epsilon')$ // Embedding of $Y_T$ with distortion

$1 + \epsilon'$

$(H_X, D_X) \leftarrow$ expand each vertex in $H_Y, D_Y$ into all hubs it represents in

$X_T$

$H_T \leftarrow H_X$ // Initially the embedding $H_T$ of $T$ is $H_X$

$D_T \leftarrow D_X$ // Initially the tree decomposition $D_T$ of $T$ is $D_X$

root($D_T$) $\leftarrow$ root($D_X$) // Set the root bag of the tree
decomposition

foreach $(H_{T'}, D_{T'}, i)$ in $\text{Towns}$ do // Join towns to $H_T$

// Find the connecting bag $b$ for $T'$

$T'' \leftarrow$ closest sibling town to $T'$ in $T$

$i \leftarrow$ level for which $\text{dist}_G(T', T'') \in (r_i, r_{i+1}]$

$h \leftarrow$ closest hub in $X_T$ to $T'$

$j \leftarrow \log_2 r_i$

$j \leftarrow$ highest level of $D_X$

$C \leftarrow$ cluster containing $h$ at level $i = \min\{j, \lceil i + \lfloor \log_2(d/\epsilon) \rfloor \}$ in

split-tree of $X_T$

$b \leftarrow$ bag in $D_X$ corresponding to cluster $C$

// Connect $T'$ to $X_T$ in the embedding

Add all vertices and edges of $H_{T'}$ to $H_T$

Add edge $(u, v)$ with length $\text{dist}_G(u, v)$ to $H_T$ for each pair $u \in T'$,

$v \in b$

// Add $D_{T'}$ to the tree decomposition $D_T$ of $H_T$

Merge $D_{T'}$ and $D_T$ by connecting root($D_{T'}$) with $b$

Add all vertices of $b$ to each bag of $D_{T'}$

Add all hubs of $X_T \cap T'$ to each bag of $D_{T'}$, and also to $b$ and all
descendants of $b$ in $D_X$ (but not the descendants of $b$ in $D_T$ that are
bags of some $D_{T''}$ for some child town $T'' \neq T'$ of $T$)

return $(H_T, D_T)$

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
that $D_X$ is a valid tree decomposition, i.e., it satisfies all properties of Definition 2.1. We will show in section 7 that the number of approximate core hubs in each child town is bounded, and therefore the growth of the treewidth caused by replacing a vertex by its represented hubs is also bounded. We also need to bound the extra distortion incurred by going from $H_Y$ to $H_X$ and show that a $1 + \varepsilon$ distortion of $H_Y$ translates into a $1 + O(\varepsilon)$ distortion of $H_X$, which entails reproving the relevant parts of Theorem 2.3.

After computing the embedding $H_X$ for $X_T$, we connect each recursively computed embedding for the child towns of $T$ (line 11 of Algorithm 1) to $H_X$ to form the final embedding $H_T$. We need to argue that $H_X$ exists every time there are child towns to connect. From Lemma 3.3 we know that $T$ has at least two child towns if it has any. In section 5 we will show (in Lemma 5.1) that there is a core hub $h$ in $T$ on any shortest path between a pair of children towns. By Theorem 4.2, there is an approximate core hub in $X_T$ close to $h$. Since $X_T$ is nonempty, $H_X$ exists. Once we compute $H_X$ we connect every vertex of a child town $T'$ to all hubs in a bag $b$ of the tree decomposition $D_X$ of $H_X$. This bag $b$ is $\log_2(d/\varepsilon)$ levels higher in the split-tree decomposition than the level corresponding to the shortest distance that needs to be bridged from $T'$ to any other vertex in $T$. At the same time we will make sure that the net defining $b$ is fine enough so that lengths of connections passing through $b$ are preserved to a sufficient degree. This way, short connections from $T'$ to core hubs with length up to $O(1/\varepsilon)$ times the separation of $T'$ are preserved in expectation by routing through the hubs in $b$. Connections to more distant hubs can be rerouted from a hub close to $T'$ through the embedding $H_X$ with only an $\varepsilon$ overhead, as we will prove in section 5.

Recall that levels of the split-tree decomposition are denoted by $i, j$, etc. To determine the level of the bag $b$, note that due to our growth rate of $c/4 = 1 + \lambda/4$ of the levels (and the assumption that the violation $\lambda$ is at most 4) the intervals $(r_i, 2r_i]$, $(r_i, r_{i+1}]$, and $[r_i, r_{i+1}]$, and let $\bar{i} = \lceil \log_2 r_i \rceil$ be the corresponding level of the split tree decomposition of $D_X$. Now let $h \in X_T$ be the closest approximate core hub to $T'$ (which might lie inside of $T'$). If $\bar{j}$ is the highest level of $D_X$, i.e., it is the level of the cluster containing all of $X_T$, then the bag $b$ of the tree decomposition $D_X$ is the one on level $\bar{l} = \min\{\bar{j}, \bar{i} + \lceil \log_2(d/\varepsilon) \rceil\}$ for which the corresponding cluster $C$ contains $h$. All edges between vertices of $T'$ and $b$ are added to the embedding for $T$ (lines 27 and 28 of Algorithm 1), and we call the bag $b$ the connecting bag for $T'$.

Note that there are several parameters $\varepsilon$ we could adjust independently: the target distortion of Talwar’s algorithm, the level in the split-tree decomposition at which a child town is attached, and the amount of adjustment permitted in defining $X_T$. The latter two parameters we set to $\varepsilon$, but the distortion in Theorem 2.3 needs to be smaller. We use $\varepsilon'$ for the target distortion of this embedding and set $\varepsilon' = \varepsilon^2$.

5. The expected distortion of the embedding. We now show that the expected distortion of the constructed embedding $H$ is $1 + O(\varepsilon)$. For this, we focus on a pair of vertices $u, v \in V$ and argue that

$$E[\text{dist}_H(u,v)] \leq (1 + O(\varepsilon))\text{dist}_G(u, v).$$

The high-level idea is rather intuitive: suppose that dist$_G(u, v) \in (r_i, r_{i+1}]$ for some $i$ and let $T \in T'$ be a town $(a)$ that contains both $u$ and $v$, and $(b)$ whose child towns separate $u$ and $v$; i.e., $u$ and $v$ are in different child towns of $T$. We first argue that there is a level-$i$ core hub $h$ of $T$ that lies on the unique shortest $u$–$v$ path.
Lemma 5.1. Let \( u \) and \( v \) be vertices that lie in different child towns of \( T \), and \( i \) be such that \( \text{dist}_G(u, v) \in (r_i, r_{i+1}] \). There is a core hub \( h \in C_i \cap \text{SPC}(r_i) \) of \( T \) on level \( i \) that hits the shortest path between \( u \) and \( v \).

Proof. By definition, \( \text{SPC}(r_i) \) must contain some hub \( h \) on the shortest \( u - v \) path. Recall that the town \( T \) can be partitioned into its core \( C_i \) on level \( i \) and the child towns on levels at least \( i \). If hub \( h \) is not a core hub, \( h \not\in \text{SPC}(r_i) \cap C_i \), then it is either outside of \( T \) or in a child town of \( T \) on a level at least \( i \).

If \( h \) lies in a child town \( T' \) of \( T \), we can assume w.l.o.g. that \( v \not\in T' \) since \( v \) and \( u \) lie in different child towns. As a hub on level \( i \), \( h \) cannot be in a town on level \( i \) by Definition 3.1, so \( T' \) is a town on level \( i + 1 \) or above. By Lemma 3.2 we then know that \( \text{dist}_G(v, h) > r_{i+1} \), but at the same time, \( \text{dist}_G(v, u) \leq r_{i+1} \)—a contradiction. If \( h \) lies outside of \( T \), then by Lemma 3.2 \( \text{dist}_G(v, u) \geq \text{dist}_G(v, h) \geq \text{dist}(T, V \setminus T) > r_j \), where \( j \) is the level of \( T \). However by the same lemma, \( \text{dist}_G(v, u) \leq \text{diam}(T) \leq r_j \)—again a contradiction. \( \square \)

By Theorem 4.2 it now follows that there is an approximate core hub \( h_X \in X_T \) such that

\[
\text{dist}_G(h, h_X) \leq \varepsilon r_i = O(\varepsilon) \text{dist}_G(u, v),
\]

since \( r_{i+1}/r_i = O(1) \) using our assumption that \( c = O(1) \). We are also able to show that the expected distances between \( u \) and \( h_X \) and \( v \) and \( h_X \), respectively, are well preserved by \( H \).

Lemma 5.2. Let \( v \) be a vertex in a child town \( T' \) of \( T \in T \), and let \( h_X \) be an approximate core hub in \( X_T \). If the distance to the closest sibling town of \( T' \) is \( r \), then \( E[\text{dist}_H(v, h_X)] \leq (1 + O(\varepsilon)) \text{dist}_G(v, h_X) + O(\varepsilon r) \).

Since \( u \) lies in a different child town than \( v \) and \( \text{dist}_G(u, v) \in (r_i, r_{i+1}] \), we get \( O(\varepsilon r) = O(\varepsilon \cdot \text{dist}_G(u, v)) \) in Lemma 5.2. Hence, using triangle inequality, the bound on the expected distance in this lemma immediately implies the following:

\[
E[\text{dist}_H(v, u)] \leq E[\text{dist}_T(v, h_X)] + E[\text{dist}_H(h_X, u)] \\
\leq (1 + O(\varepsilon)) \text{dist}_G(v, h_X) + (1 + O(\varepsilon)) \text{dist}_G(h_X, u) + O(\varepsilon \cdot \text{dist}_G(u, v)) \\
\leq (1 + O(\varepsilon)) (\text{dist}_G(v, u) + \text{dist}_G(h_X, h_X) + \text{dist}_G(h_X, h) + \text{dist}_G(h, u)) \\
+ O(\varepsilon) \text{dist}_G(u, v) \\
\leq (1 + O(\varepsilon)) \text{dist}_G(v, u),
\]

where the last equality uses the fact that \( h_X \) lies close to a shortest \( u, v \)-path (see (1)). Together with the fact that \( \text{dist}_G(u, v) \leq \text{dist}_H(u, v) \), this implies our stretch bound.

Theorem 5.3. The expected stretch of the embedding \( H \) of \( G \) is \( 1 + O(\varepsilon) \).

The remainder of this section is devoted to providing a proof of Lemma 5.2, for which we will need some further details from Talwar’s embedding of low doubling metrics into bounded treewidth graphs.

5.1. The distortion of an embedding for approximate core hubs. Before proceeding with the proof of Lemma 5.2 we will first need to have a closer look at the properties of Talwar’s split-tree decomposition. We will use these properties to prove that our computed embedding \( H_X \) of the approximate core hubs \( X_T \) has distortion \( 1 + O(\varepsilon) \).

Lemma 5.4 (see [38]). The split-tree decomposition for a metric \( (X, \text{dist}) \) with doubling dimension \( d \) and aspect ratio \( \alpha \) satisfies the following properties:

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
EMBEDDING HIGHWAY DIMENSION INTO BOUNDED TREEWIDTH

(1) there are $\log_2 \alpha + 2$ levels,
(2) the clusters on each level $i$ partition $X$,
(3) the diameter of a cluster at level $i$ is at most $2^{i+1}$, and
(4) the probability that any points $x, y \in X$ are in distinct level $i$ clusters is $O(d \cdot \text{dist}(x, y)/2^i)$.

Recall the notion of $\delta$-net from Definition 2.4. The main result of Talwar [38] that we use for our embedding is the following more detailed account of Theorem 2.3.

**Theorem 5.5** (see [38]). Let $(X, \text{dist})$ be a metric with doubling dimension $d$ and aspect ratio $\alpha$. In polynomial time we can compute a probabilistic embedding $D$ of $X$ into bounded treewidth graphs. In particular, a computed graph $H \in D$ has a tree decomposition $D$ with the following properties:

(i) each bag $b$ in $D$ corresponds to a cluster $C$ in the split-tree decomposition of $(X, \text{dist})$, and the tree underlying $D$ is precisely that of the split-tree decomposition;

(ii) the nets of the clusters form a hierarchy, i.e., every vertex in a bag $b$ is also contained in one of the children of $b$ in the tree $D$;

(iii) a bag $b$ corresponding to a cluster $C$ at level $i$ consists of a $\beta 2^i$-net of $C$ for some $\beta > 0$; and

(iv) using a $\beta 2^i$-net for clusters at level $i$, the expected distortion of $H$ is $1 + O(\beta \log \alpha)$, and the treewidth of $D$ is at most $O(1/\beta)$.

In particular there is a $\beta = \Theta(\varepsilon'/(d \log \alpha))$ such that the expected distortion is $1 + \varepsilon'$, and the treewidth is $(d \log \alpha)/\varepsilon' \cdot O(1/\beta)$.

For every bag $b$ in $D$, the graph $H$ contains a complete graph on the nodes in $b$. The $\beta 2^i$-net in each bag serves as a set of portals, through which connections leaving the cluster are routed, analogous to those in [8]. The bound on the stretch follows from Lemma 5.4. (See [38] for the details.) The bound on the treewidth follows from the fact that a $\beta 2^i$-net in a cluster with diameter at most $2^{i+1}$ has aspect ratio $O(1/\beta)$ and the following property of low doubling dimension metrics.

**Lemma 5.6** (see [30]). Let $(X, \text{dist})$ be a metric with doubling dimension $d$ and $Y \subseteq X$ be a set with aspect ratio $\alpha$. Then $|Y| \leq 2d[\log_2 \alpha]$.

To analyze the distortion of the embedding $H_X$, we rely on the following useful fact that relates properties of hubs in $X_T$ and their representatives in $Y_T$. Recall that a cluster $C_X$ of $X_T$ is formed from a cluster $C_Y$ of $Y_T$ by expanding each hub $h \in C_Y$ into all vertices in $X_T$ that $h$ represents, and a bag $b_X$ of the tree decomposition $D_X$ of $X_T$ is formed by the same procedure from a bag $b_Y$ of the tree decomposition $D_Y$ of $Y_T$. For such pairs of clusters and bags we obtain the following.

**Lemma 5.7.** If $b_Y$ is a $\delta$-net of $C_Y$ for some $\delta$, then $b_X$ is a $2\delta$-cover of $C_X$, i.e., for each $h_X \in C_X$ there is an $h_Y \in b_X$ such that $\text{dist}_G(h_X, h_Y) \leq 2\delta$.

**Proof.** Let $h_X \in C_X$. If $h_X \in b_X$, we are done. If not, let $h_Y$ be $h_X$’s representative in $Y_T$, and let $T'$ be the child town of $T$ for which $h_X, h_Y \in X_T \cap T'$. We obtained $b_X$ by expanding each $h \in b_Y$ into all vertices it represents, so $h_X \notin b_X$ implies $h_Y \notin b_Y$. Let $h_Y' \in b_Y$ be the closest vertex in $b_Y$ to $h_Y$. The set $b_Y$ is a $\delta$-net of $C_Y$, so $\text{dist}_G(h_Y, h_Y') \leq \delta$, but $h_Y' \notin T'$, since $h_Y' \neq h_Y$, and each town contains at most one representative. By Lemma 3.2, $\text{diam}(T') \leq \text{dist}_G(T', V \setminus T')$, so $\text{dist}_G(h_X, h_Y) \leq \text{dist}(h_Y, h_Y')$, which means that $\text{dist}_G(h_X, h_Y') \leq 2\delta$. Finally, $h_Y' \in b_X$, since $b_Y \subseteq b_X$. \[ \square \]
Another useful tool is given by the following lemma, which compares the separation probabilities of approximate core hubs and their representatives.

**Lemma 5.8.** Let $u, v \in X_T$ be two hubs with respective representatives $u', v' \in Y_T$. If $u' \neq v'$, then the probability with which $u$ and $v$ are in distinct level $i$ clusters is $O(d \cdot \text{dist}_G(u,v)/2^i)$, where $d$ is the doubling dimension of $Y_T$.

**Proof.** If the representatives $u'$ and $v'$ of $u$ and $v$ differ, then $u$ and $v$ must lie in different child towns $T'$ and $T''$ of $T$. By Lemma 3.2, $\text{diam}(T') < \text{dist}_G(T',V \setminus T') \leq \text{dist}_G(T',T'')$, so that $\text{dist}_G(u,u') \leq \text{dist}_G(u,v)$, and similarly for $\text{dist}_G(v,v')$. Hence $\text{dist}_G(u',v') \leq \text{dist}_G(u',u) + \text{dist}_G(u,v) + \text{dist}_G(v,v') \leq 3 \cdot \text{dist}_G(u,v)$. By Lemma 5.4(4), the separation probability of $u'$ and $v'$ on level $i$ is at most $O(d \cdot \text{dist}_G(u',v')/2^i)$. Since $u$ and $v$ lie in different clusters if and only if their representatives do, the probability of $u$ and $v$ being separated is $O(d \cdot \text{dist}_G(u,v)/2^i)$. \qed

The next lemma bounds the distortion of $H_X$. Its proof closely mirrors Talwar’s proof of Theorem 5.5 (c.f. [38]).

**Lemma 5.9.** If the embedding $H_Y$ of $(Y_T,\text{dist}_G)$ is computed according to Theorem 5.5, then the constructed embedding $H_X$ of $(X_T,\text{dist}_G)$ has expected distortion $1 + O(\varepsilon')$.

**Proof.** Consider a cluster $C_Y$ on level $\bar{i}$ in the split-tree decomposition of $Y_T$ given by Lemma 5.4. For any $h \in C_Y$ the $i$-parent of $h$ is the closest vertex to $h$ in the bag $b_Y$ corresponding to $C_Y$. Since by Theorem 5.5 the bag $b_Y$ consists of a $\beta 2^i$-net of $C_Y$, the distance between $h$ and its $i$-parent is at most $\beta 2^i$. Let $C_X$ be the cluster in $X_T$ formed by expanding each $h \in C_Y$ into all vertices in $X_T$ that $h$ represents, and let $b_X$ be the corresponding bag formed by the same procedure from $b_Y$. We define the $i$-parent of a vertex $w \in C_X$ in the same way as for $C_Y$, i.e., it is the closest vertex to $w$ in $b_X$. According to Lemma 5.7, the distance from $w$ to its $i$-parent is at most $2 \beta 2^i$.

For an arbitrary pair $u, v \in X_T$ we bound the distortion of their distance in $H_X$ by considering the path along the $i$-parents of $u$ and $v$ for increasing values of $i$. More concretely, since the bags of the tree decomposition $D_Y$ of $H_Y$ form a hierarchy by Theorem 5.5, the same is true for the bags of the tree decomposition $D_X$ of $H_X$. Thus on the lowest level $\bar{l}$ of the split-tree decomposition, the $\bar{l}$-parent of a vertex $w$ is $w$ itself. We inductively define $v_{\bar{l}} = v$, $u_{\bar{l}} = u$, and $v_i$ and $u_i$ to be the $i$-parent of $v_{i-1}$ and $u_{i-1}$, respectively, for any level $\bar{l} > i$. Since the bags of $D_X$ form a hierarchy, for each level $i > \bar{l}$ the edges $\{u_{i-1}, u_i\}$ and $\{v_{i-1}, v_i\}$ exist in $H_X$. Thus the distance from $u_{i-1}$ to $u_i$ and from $v_{i-1}$ to $v_i$ is at most $2 \beta 2^i$ in $H_X$. Now, let $\bar{j}$ be the lowest level at which $u$ and $v$ lie in the same cluster of $X_T$. In particular, the $\bar{j}$-parents $v_{\bar{j}}$ and $u_{\bar{j}}$ lie in the same bag of $D_X$, and so there is an edge $\{u_{\bar{j}}, v_{\bar{j}}\}$ in $H_X$. We next bound the expected length of the path $P = (u = u_{\bar{l}}, u_{\bar{l}+1}, \ldots, u_j, v_j, v_{j-1}, \ldots, v_i = v)$ in $H_X$ in terms of $\text{dist}_G(u,v)$.

For this we need to bound the probability with which any pair of $i$-parents $u_i$ and $v_i$ lie in different clusters of $X_T$ on level $\bar{i}$. Note that $u$ and $v$ always lie in the same cluster as their respective $i$-parent, and so $u_{\bar{i}}$ and $v_{\bar{i}}$ lie in different clusters of $X_T$ on level $\bar{i}$ if and only if $u$ and $v$ lie in different clusters of $X_T$ on the same level. Lemma 5.4 gives a bound for the probability with which representatives lie in different clusters of $Y_T$ in terms of the distance between them. Let $u', v' \in Y_T$ be the respective representatives of $u$ and $v$. If $u' = v'$, then obviously $\text{dist}_G(u',v') = 0$. Otherwise, $u'$ and $v'$ lie in different child towns of $T$. By Lemma 5.8, this means that $u$ and $v$ lie in different clusters on level $\bar{i}$ with probability $O(d \cdot \text{dist}_G(u,v)/2^i)$. Let $A_i$ be the indicator variable that is 1

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
if \( u_i \) and \( v_i \) lie in different clusters of \( X_T \) on level \( \bar{l} \), and 0 otherwise, so that \( \Pr[A_i = 1] = O(d \cdot \text{dist}_G(u,v)/2^l) \).

Consider the subpaths of \( P \) from \( u \) to \( u_j \) and \( v \) to \( v_j \). The length of each such path is at most \( \sum_i 2\beta2^{i+1}A_i \). Accordingly, the edge \( \{u_j, v_j\} \) has length at most \( \text{dist}_G(u,v) + 2 \sum_i 2\beta2^{i+1}A_i \). Since there are at most \( \log_2 \alpha \) levels in the split-tree decomposition, we can bound the expected length of \( P \) by

\[
\text{dist}_G(u,v) + 4 \sum_{i=\bar{l}}^{\log_2 \alpha} 2\beta2^{i+1} \cdot O(d \cdot \text{dist}_G(u,v)/2^l)
\]

\[
= (1 + O(\beta d \log \alpha))\text{dist}_G(u,v) = (1 + O(\epsilon'))\text{dist}_G(u,v),
\]

where we use that \( \beta = O(\epsilon'/(d \log \alpha)) \) by Theorem 5.5.

5.2. The distortion of the embedding of the graph. We now turn to proving Lemma 5.2. For this, throughout this section, we focus on a town \( T \) of the towns decomposition \( T \). We further let \( T' \) be some child town of \( T \), and we let the distance \( \bar{r} \) between \( T' \) and the closest sibling town be in the interval \( [r_i, r_{i+1}] \). Further, we define \( b \) to be the connecting bag of \( T' \) (c.f. Algorithm 1), and let \( C \) be the corresponding cluster in the split-tree decomposition of the approximate core hubs \( X_T \).

Given vertex \( v \in T' \subseteq T \), and some core hub \( h_X \in X_T \), the goal is to bound their expected distance in the constructed embedding \( H \) in terms of their distance in the input graph \( G \). If \( H \) contains an edge between \( v \) and \( h_X \), then we are of course immediately done, but this may not be the case. For example, in the construction of the embedding, we add direct links between vertices of \( T' \) and members (i.e., net points) of the connecting bag \( b \), but \( h_X \) may not be a member of \( b \). We first consider this issue and show that, even if \( h_X \in C \setminus b \), then \( b \) at least contains a net point close to \( h_X \).

**Lemma 5.10.** For any approximate core hub \( h \in X_T \cap C \), the bag \( b \) contains a net point \( w \) such that \( \text{dist}_H(h,w) = O(\varepsilon r_i) \).

**Proof.** Let \( \bar{l} \) be the level of \( b \) in the tree decomposition \( D_X \), which by Algorithm 1 is at most \( \bar{l} = \bar{l} + \log_2(d/\varepsilon) \), where \( \bar{l} = \log_2 r_i \). If \( h \in b \), there is nothing to show. By (ii) of Theorem 5.5, the bags of \( D_Y \) form a hierarchy, which by construction of \( D_x \) means that the bags of \( D_Y \) do too. Thus \( h \notin b \) is a vertex in a bag on some level below \( \bar{l} \), and so we can reach some vertex of \( b \) from \( h \) in \( H_X \) by starting at the bag containing \( h \) and following the edges to higher level bags until we reach \( b \). More concretely, the bags computed for the tree decomposition \( D_Y \) of the representative hubs \( Y_T \) contain \( \beta \) cover points that also bound the corresponding clusters by Theorem 5.5. Hence by Lemma 5.7, the bags of \( D_X \) contain \( 1/\beta \)-nets of the corresponding clusters by Theorem 5.5. The above lemma provides a vertex \( w \) of the connecting bag \( b \) of \( T' \) through which we can connect to a hub \( h_X \), if \( h_X \in C \). In the case \( h_X \) lies outside of \( C \), however, as we will see the following lemma provides such a vertex in \( b \) to connect to \( h_X \).

**Lemma 5.11.** For any \( v \in T' \) and approximate core hub \( h_X \in X_T \setminus T' \), the connecting bag \( b \) of \( T' \) contains a vertex \( w \) such that \( \text{dist}_G(v,w) = O(\text{dist}_G(v,h_X)) \) and \( \text{dist}_G(v,w) = O(r_i) \).

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Proof. Recall that, by our choice in Algorithm 1, cluster \( C \) corresponding to connecting bag \( b \) of \( T' \) contains the closest hub \( h \in X_T \) to \( T' \). By Lemma 5.10, there exists \( w \in b \) with \( \text{dist}_G(h,w) \leq \text{dist}_H(h,w) = O(\varepsilon r_i) \) (cf. Figure 4). As by triangle inequality \( \text{dist}_G(v,w) \leq \text{dist}_G(v,h) + \text{dist}_G(h,w) \), it remains to show that \( \text{dist}_G(v,h) = O(r_i) \) in order to prove \( \text{dist}_G(v,w) = O(r_i) \), if \( \varepsilon \) tends to zero. By Lemma 5.1 there is a core hub \( u \) of \( T \) on level \( i \), which lies on the shortest path between \( T' \) and \( T'' \), the closest sibling town to \( T' \), and thus \( u \) is at most as far from \( T' \) as any vertex in \( T'' \). Hence \( \text{dist}_G(T',u) \leq r_{i+1} \), since we assumed that the distance \( r \) between \( T' \) and \( T'' \) lies in the interval \((r_i,r_{i+1})\]. By Theorem 4.2 there is an approximate core hub \( v' \in X_T \) for which \( \text{dist}_G(u,v') \leq \varepsilon r_i \). Hence the closest approximate core hub \( h \) is at distance at most \( r_{i+1} + \varepsilon r_i \) from \( T' \). From Lemma 3.2 it follows that every town on level at least \( i+1 \) has distance more than \( r_{i+1} \) to any other town. Since the distance \( r \) from \( T' \) to \( T'' \) is at most \( r_{i+1} \), the level of \( T'' \) is at most \( i \). Hence the same lemma also implies that the diameter of \( T'' \) is at most \( r_i \) and thus \( \text{dist}_G(v,h) \leq \text{diam}(T') + \text{dist}_G(T',h) \leq r_{i+1} + (1+\varepsilon) r_i = O(r_i) \), since \( r_{i+1}/r_i = c \) is constant and we assume that \( \varepsilon \) tends to zero. This implies \( \text{dist}_G(v,w) = O(r_i) \) as claimed. Note that since \( h_X \) lies outside of \( T' \), \( \text{dist}_G(v,h_X) \geq \text{dist}_G(T',V \setminus T') > r_i \) by Lemma 3.2, which immediately implies the remaining bound \( \text{dist}_G(v,w) = O(\text{dist}_G(v,h_X)) \).

So far we have identified vertices \( w \) in the connecting bag \( b \) through which we are able to connect to a hub \( h_X \) from a vertex \( v \in T' \) for the two cases when \( h_X \in C \) and \( h_X \notin C \). The next lemma provides a bound on the probability with which we need to consider each of these cases. Additionally it also bounds the distance from \( v \) to \( h_X \) in the former case.

**Lemma 5.12.** Let \( h_X \) be an approximate core hub in \( X_T \), and \( v \in T' \), and then \( \Pr[h_X \notin C] = O(\varepsilon \cdot \text{dist}_G(v,h_X)/r_i) \). In addition, \( \text{dist}_H(v,h_X) \leq \text{dist}_G(v,h_X) + O(\varepsilon r_i) \) if \( h_X \in C \).

**Proof.** If \( h_X \in T' \), then \( h_X \in C \), since by Algorithm 1 the cluster \( C \) contains the closest approximate core hub to \( T' \) and all hubs of \( X_T \) that are represented by the same hub of \( Y_T \cap C \) (i.e., that are of the same child town) are contained in \( C \). Hence if \( h_X \notin C \), then \( h_X \notin T' \). Consider the vertex \( v \in b \) for which \( \text{dist}_G(v,w) = O(\text{dist}_G(v,h_X)) \), which now exists due to Lemma 5.11. The hub \( h_X \) is in \( C \) if and only if \( w \) and \( h_X \) are in the same cluster on the level \( \tilde{l} \) of \( C \). If the level \( \tilde{l} \) of the cluster \( C \) is the level \( j \) of the root of \( D_X \), then \( C \) contains all vertices of \( T \) including \( h_X \) and \( w \), and so if \( h_X \notin C \), then \( \tilde{l} \neq j \). If \( w \) and \( h_X \) have the same representative, they will be in the same cluster by Algorithm 1, so that if \( h_X \notin C \), then \( w \) and \( h_X \) have different representatives in \( Y_T \).

By these observations, the probability with which \( w \) and \( h_X \) lie in different clusters is \( O(d \cdot \text{dist}_G(w,h_X)/2^l) \) using Lemma 5.8, which in turn can be bounded.
EMBEDDING HIGHWAY DIMENSION INTO BOUNDED TREEWIDTH

by $O(\varepsilon \cdot \text{dist}_G(w, h_X)/r_i)$, as $2^l = \Theta(dr_i/\varepsilon)$ by Algorithm 1 whenever $l \neq j$. Upper bounding $\text{dist}_G(w, h_X)$ in terms of $\text{dist}_G(v, v) + \text{dist}_G(v, h_X) = O(\text{dist}_G(v, h_X))$ we obtain $\Pr[h_X \notin C] = O(\varepsilon \cdot \text{dist}_G(v, h_X)/r_i)$.

To bound the distance if $h_X \in C$, by Lemma 5.10 we know that there is a vertex $h_b \in b$ such that $\text{dist}_H(h_b, h_X) = O(\varepsilon r_i)$, and $v$ has an edge in $H$ to $h_b$. Therefore $\text{dist}_H(v, h_b) \leq \text{dist}_H(v, h_b) + \text{dist}_H(h_b, h_X) = \text{dist}_G(v, h_b) + O(\varepsilon r_i)$. Since $\text{dist}_C(h_b, h_X) \leq \text{dist}_H(h_b, h_X)$, we can upper bound $\text{dist}_G(v, h_b)$ by $\text{dist}_C(v, h_b) + \text{dist}_H(h_b, h_b)$, which proves the claim. 

Lemma 5.12 provides a bound on the distance between vertices of $T'$ and approximate core hubs in $C$. We also need to bound the distance between vertices of $T'$ and core hubs of $T$ that are not in $C$. The following lemma will be useful in this endeavor.

**Lemma 5.13.** Let $H_X$ be the probabilistic embedding of $(X_T, \text{dist}_G)$ with expected distortion $1 + O(\varepsilon')$ given by Lemma 5.9. Let $x, y \in X_T$, and let $C$ be a cluster in the randomized split-tree decomposition containing $x$. Then $E[\text{dist}_{H_X}(x, y) \mid y \notin C] \leq (1 + O(\varepsilon')) \cdot \text{dist}_G(x, y)$.

**Proof.** By Lemma 5.9, the expected distance between $x$ and $y$ in $H$ is at most $(1 + O(\varepsilon'))$ times their distance in metric $(X_T, \text{dist}_G)$, and hence

$$E[\text{dist}_{H_X}(x, y)] = \Pr[y \notin C] \cdot E[\text{dist}_{H_X}(x, y) \mid y \notin C] + \Pr[y \in C] \cdot E[\text{dist}_{H_X}(x, y) \mid y \in C] \leq (1 + O(\varepsilon')) \cdot \text{dist}_G(x, y).$$

Embedding $H_X$ dominates $(X_T, \text{dist}_G)$, and hence $E[\text{dist}_{H_X}(x, y) \mid y \in C] \geq \text{dist}_G(x, y)$. The inequality above therefore implies that

$$\Pr[y \notin C] \cdot E[\text{dist}_{H_X}(x, y) \mid y \notin C] + (1 - \Pr[y \notin C]) \cdot \text{dist}_G(x, y) \leq (1 + O(\varepsilon')) \cdot \text{dist}_G(x, y).$$

Rearranging, $\Pr[y \notin C] \cdot E[\text{dist}_{H_X}(x, y) \mid y \notin C] - \text{dist}_G(x, y) \leq O(\varepsilon') \cdot \text{dist}_G(x, y)$, and

$$E[\text{dist}_{H_X}(x, y) \mid y \notin C] \leq \left(1 + \frac{O(\varepsilon')}{\Pr[y \notin C]}\right) \cdot \text{dist}_G(x, y).$$

We are now ready to bound the distance between a vertex $v \in T$ and any core hub in $H_T$, given the tools of the above lemmas.

**Proof of Lemma 5.2.** Let $C$ be the cluster corresponding to the connecting bag $b$ of $T'$. We bound $E[\text{dist}_H(v, h_X)]$ in terms of the conditional expected values $E[\text{dist}_H(v, h_X) \mid h_X \in C]$ and $E[\text{dist}_H(v, h_X) \mid h_X \notin C]$. If $h_X \in C$, we get a (deterministic) bound on the distance between $v$ and $h_X$ from Lemma 5.12. Hence $E[\text{dist}_H(v, h_X) \mid h_X \in C] \leq \text{dist}_G(v, h_X) + O(\varepsilon r_i)$. If $h_X \notin C$, then $h_X$ contains the closest hub to $T'$ and all hubs of $X_T$ in the same child town of $T$ end up in the same cluster after expanding all hubs of $G_T$ into the hubs of $X_T$ that they represent. Hence if $h_X \notin C$, then $h_X \notin T'$, and by Lemma 5.11 there is a vertex $w \in b$ for which $\text{dist}_G(v, w) = O(\text{dist}_G(v, h_X))$. Both $w$ and $h_X$ are approximate core hubs, and so $E[\text{dist}_H(w, h_X) \mid h_X \notin C] \leq E[\text{dist}_{H_X}(w, h_X) \mid h_X \notin C]$, as $H$ contains $H_X$. Applying Lemma 5.13 on this
conditionally expected distance, we obtain

\[
\Pr[h_X \notin C | E[\text{dist}_H(v, h_X) | h_X \notin C]] \\
\leq \Pr[h_X \notin C] \left( \text{dist}_G(v, w) + E[\text{dist}_H(w, h_X) | h_X \notin C] \right) \\
\leq \Pr[h_X \notin C] \left( \text{dist}_G(v, w) + \left( 1 + \frac{O(\varepsilon')}{\Pr[h_X \notin C]} \right) \text{dist}_G(w, h_X) \right) \\
= \Pr[h_X \notin C] \left( \text{dist}_G(v, w) + \text{dist}_G(w, h_X) \right) + O(\varepsilon') \text{dist}_G(w, h_X) \\
+ O(\varepsilon') \left( \text{dist}_G(w, v) + \text{dist}_G(v, h_X) \right) \\
= \Pr[h_X \notin C] \left( \text{dist}_G(v, w) + \text{dist}_G(v, h_X) \right) + O(\varepsilon' \cdot \text{dist}_G(v, h_X)).
\]

From Lemma 5.11 we also know that dist\(_G(v, w) = O(r_i)\). Additionally using that \(\varepsilon' = \varepsilon^2\), and the bound on \(\Pr[h_X \notin C]\) in Lemma 5.12, the expression above is

\[
\Pr[h_X \notin C] \text{dist}_G(v, h_X) + O\left( \varepsilon \frac{\text{dist}_G(v, h_X)}{r_i} \right) O(r_i) + O(v' \cdot \text{dist}_G(v, h_X)) \\
= \Pr[h_X \notin C] \text{dist}_G(v, h_X) + O(\varepsilon) \text{dist}_G(v, h_X).
\]

Combining the above bounds we obtain

\[
E[\text{dist}_H(v, h_X)] \\
= \Pr[h_X \in C] E[\text{dist}_H(v, h_X) | h_X \in C] + \Pr[h_X \notin C] E[\text{dist}_H(v, h_X) | h_X \notin C] \\
\leq \Pr[h_X \in C] \left( \text{dist}_G(v, h_X) + O(\varepsilon r_i) \right) + \Pr[h_X \notin C] \text{dist}_G(v, h_X) \\
+ O(\varepsilon) \text{dist}_G(v, h_X) \\
= (1 + O(\varepsilon)) \text{dist}_G(v, h_X) + O(\varepsilon r_i),
\]

where \(r_i = \Theta(r)\), which proves the claim.

6. The doubling dimension of approximate core hubs. The aim of this section is to give a proof of Theorem 4.2 by showing that for any town \(T \in T\) there is a set \(X_T \subseteq T\) of approximate core hubs with bounded doubling dimension. We first define the set \(X_T\) and then compare its properties with those of the core hubs. In particular, even though we obtain the approximate core hubs by shifting the core hubs to positions nearby, the resulting set is still locally sparse on each level. In addition, they are also locally nested. Roughly speaking, this means that within a small ball of radius \(\varepsilon r_i\) for some level \(i\), all approximate core hubs above level \(i\) are “nested,” i.e., contained in one another. This property will help us in bounding the doubling dimension of \(X_T\) independently of the aspect ratio.

The set \(X_T\) of a town \(T\) of level \(j\) is the union of sets \(X_T^i\), one for each level \(i \in \{1, \ldots, j - 1\}\), which are defined inductively as follows in Algorithm 2. We call a vertex in \(X_T^i\) an approximate core hub of \(T\) on level \(i\). Recall that \(C_i\) is the core of \(T\) at level \(i\) (Definition 4.1), and \(C_0 = \emptyset\) since the sprawl is empty on level 0.
**Algorithm 2:** Defining $X_T$

1. $X_T^1 \leftarrow C_1 \cap \text{spc}(r_1)$
2. for $i = 2, \ldots, j - 1$
3.     $X_T^i \leftarrow \emptyset$
4.     foreach $h \in C_i \cap \text{spc}(r_i)$ do
5.         if $\exists h' \in X_T^l$ for some $l < i$ such that $\text{dist}(h, h') \leq \varepsilon r_i$ then add $h'$ to $X_T^i$
6.         else add $h$ to $X_T^i$
7. return $\bigcup_{i=1}^{j-1} X_T^i$

Note that this definition of $X_T$ fulfills the two properties of Theorem 4.2 that there must be an approximate core hub $h' \in X_T$ within distance $\varepsilon r_i$ of each core hub $h$ of level $i$ and that $X_T$ can be computed in polynomial time. Note also that $X_T^1 \subseteq \bigcup_{i=1}^{j-1} C_i \cap \text{spc}(r_i)$, and hence the vertices in $X_T^1$ are core hubs, but not necessarily core hubs of level $i$. The main benefit of shifting core hubs to approximate core hubs is that for any town $T \in \mathcal{T}$ on level $j$, the set system $\{X_T^i\}_{i=1}^j$ is locally nested as we explain in the following lemma.

**Lemma 6.1.** Let $B$ be a set of diameter at most $\varepsilon r_i$ for some level $l$, and let $i$ be the lowest level for which $X_T^i \cap B \neq \emptyset$. The approximate core hubs on level $q \geq \max\{i, l\}$ in $B$ must also be core hubs on some level at most $\max\{l, i\}$; i.e., $B \cap X_T^q \subseteq \bigcup_{p=1}^{\max\{l, i\}} X_T^p$.

**Proof.** The statement is trivially true for $q = \max\{l, i\}$. Consider any higher level $q > \max\{l, i\}$. Since the diameter of $B$ is at most $\varepsilon r_i \leq \varepsilon r_q$ and $X_T^q \cap B \neq \emptyset$, for every $h \in B \cap C_q \cap \text{spc}(r_q)$ there is a vertex $h' \in X_T^q$ at distance at most $\varepsilon r_q$ from $h$. Hence by the definition of the approximate core hubs in Algorithm 2, $X_T^q \cap B \subseteq \bigcup_{p=1}^{q-1} X_T^p$, and the claim follows by induction. □

The cost of using approximate core hubs is that it is not immediately clear why the vertices in $X_T^i$ should still be locally sparse. This requires a tricky argument that we turn to now. The crucial observation leading to this result is that we can bound the number of hubs of a shortest path cover $\text{spc}(r_i)$ not only in a ball $B_{\varepsilon r_i/2}(v)$ using the local sparsity but also close to the ball. The approximate core hubs in $X_T^i$ are obtained by shifting the core hubs of level $i$ to lower level core hubs at distance at most $\varepsilon r_i$. Hence the number of vertices of $X_T^i \cap B_{\varepsilon r_i/2}(v)$ can be bounded by the total number of level $i$ core hubs that are within distance $\varepsilon r_i$ of $B_{\varepsilon r_i/2}(v)$. The definition of highway dimension (Definition 1.1) allows us to get a handle on the hubs in larger balls of radius $c r_i$, and this, combined with the minimality of our shortest path cover, allows us to bound the number of nearby core hubs. Specifically, in a graph of highway dimension $k$, and given a locally $s$-sparse shortest path cover, we are able to show that the approximate core hubs $X_T^i$ of level $i$ are locally $3ks$-sparse as long as the stretch parameter $\varepsilon$ is chosen to be at most 2. The lemma is stated in a slightly more general form than we need it here, since we will reuse it later.

**Lemma 6.2.** For a metric $(V, \text{dist}_G)$ induced by an underlying graph $G$ of highway dimension $k$, let $B_{\varepsilon r_i/2}(v)$ be a ball of radius $\varepsilon r_i/2$ centered at $v \in V$, and let $\text{spc}(r)$ be a minimal locally $s$-sparse shortest path cover. There are at most $3sk$ hubs $h \in \text{spc}(r)$ for which $\text{dist}_G(h, B_{\varepsilon r_i/2}(v)) \leq \varepsilon r_i/2$.

We note that this lemma does not bound the number of hubs in $\text{spc}(r)$ that lie in a ball $B_{\varepsilon r_i}(v)$, and in fact the number of hubs in $B_{\varepsilon r_i}(v) \cap \text{spc}(r)$ can be unbounded:
in a star with edges of length \( cr \) a minimal shortest path cover \( spc(r) \) may contain all vertices except the center vertex \( v \) of the star. This shortest path cover is also locally 1-sparse, since any ball of radius \( cr/2 \) contains only one vertex of the star. However the ball of radius \( cr \) centered at \( v \) contains the whole star, and thus all hubs from \( spc(r) \), i.e., a potentially unbounded number.

Since the hubs considered in Lemma 6.2 may lie outside of \( B_{cr/2}(v) \), we need to use Definition 1.1, which bounds the number of hubs in larger balls of radius \( cr \). However, the hubs given by Definition 1.1 do not necessarily coincide with those of \( spc(r) \). Therefore, we need an additional tool, as given by the following technical lemma, which relates the hubs given by Definition 1.1 with those in \( spc(r) \).

In the following lemma, we consider once more a metric induced by graph \( G = (V, E) \) of highway dimension \( k \). As usual, we let \( spc(r) \) denote a locally \( s \)-sparse shortest-path cover for radius \( r \). Consider radii \( r, \tilde{r} \) such that \( \tilde{r} < cr/2 \), and let \( B_{\tilde{r}}(v) \) be a ball of radius \( \tilde{cr} \) centered at \( v \). For each vertex \( h \in B_{\tilde{r}}(v) \cap spc(r) \), we let \( P_h \) be a shortest path that (a) lies in \( B_{\tilde{r}}(v) \), i.e., \( V(P_h) \subseteq B_{\tilde{r}}(v) \), (b) has length in \((\tilde{r}, cr/2]\), and (c) contains \( h \). If no such path exists, we let \( P_h = \perp \).

**Lemma 6.3.** Let \( W \) be the set of all vertices \( h \in B_{\tilde{r}}(v) \cap spc(r) \) for which \( P_h \neq \perp \).

Then \(|W| \leq sk\).

**Proof.** The proof follows directly from Definition 1.1. The definition implies that there is a set \( K \subseteq B_{\tilde{r}}(v) \) of at most \( k \) vertices covering all shortest paths in \( B_{\tilde{r}}(v) \) of length more than \( \tilde{r} \). In particular these vertices cover each path \( P_h \) for \( h \in W \). We have \( h \in V(P_h) \) and the length of \( P_h \) is at most \( cr/2 \), so \( dist_G(h, K) \leq cr/2 \). Therefore \( W \) can be covered by at most \( k \) balls of radius \( cr/2 \) centered at each vertex in \( K \). The set \( spc(r) \), and with that also \( W \), is locally \( s \)-sparse, so each of these balls contains at most \( s \) nodes, yielding \(|W| \leq sk\). \( \Box \)

We now prove Lemma 6.2. For this, define

\[ W = \{ h \in spc(r) \mid dist_G(h, B_{cr/2}(v)) \leq cr/2 \} \]

as the set of hubs near \( v \) whose size we want to bound. In order to accomplish this, we carefully choose three radii \( \tilde{r}_i \), where \( i \in \{1, 2, 3\} \), and let \( W_i \) be the corresponding set of hubs as defined in Lemma 6.3 (see Figure 5). We will then show that

\[ W \subseteq W_1 \cup W_2 \cup W_3 \]

and conclude that \( W \) has at most \( 3sk \) elements directly from Lemma 6.3.

**Proof of Lemma 6.2.** We first apply Lemma 6.3 for \( \tilde{r}_1 = r \) and infer that the set \( W_1 \) of hubs \( h \in spc(r) \) that cover a shortest path contained in \( B_{cr}(v) \) and with length in \((r, cr/2]\) is at most \( sk \).

Observe that, by the inclusionwise minimality of \( spc(r) \), each \( h \in spc(r) \) must hit some shortest path \( Q_h \) with length in \((r, cr/2]\). For \( h \in W \setminus W_1 \) this path \( Q_h \) is not contained in \( B_{cr}(v) \). Let \( w_h \) be a vertex on path \( Q_h \) of maximum distance from \( v \), which by assumption must lie outside the ball \( B_{cr}(v) \). We know \( dist_G(h, w_h) \leq cr/2 \), as the distance between \( h \) and \( w_h \) is bounded by the maximum length of \( Q_h \). Also let \( u_h \) be the closest vertex in \( B_{cr/2}(v) \) to \( h \). By the definition of \( W \), \( dist_G(u_h, h) \leq cr/2 \).

Since \( h \) does not cover any shortest path inside \( B_{cr}(v) \) with length in \((r, cr/2]\), we must have \( dist_G(u_h, h) \leq r \). Combining these, the distance from \( v \) to \( w_h \) is at most

\[ dist_G(v, u_h) + dist_G(u_h, h) + dist_G(h, w_h) \leq cr/2 + r + cr/2 = (c + 1)r = c(1 + 1/c)r. \]
Hence, $Q_h$ lies in the ball $B_{c\tilde{r}/2}(v)$ if we choose $\tilde{r}_2 = (1 + 1/c)r$. Furthermore, $h \in \tilde{W}_2$ if $Q_h$ has length in the interval $(\tilde{r}_2, cr/2]$.

Finally, let us consider a hub $h \in W \setminus (\tilde{W}_1 \cup \tilde{W}_2)$, for which the length of the path $Q_h$ must lie in the interval $(r, \tilde{r}_2] = (r, (1 + 1/c)r]$. Let $u_h$ and $w_h$ be defined as before. The distance between $h$ and $w_h$ is now at most $(1 + 1/c)r$, while the distance between $u_h$ and $w_h$ is more than $cr/2$, as $u_h \in B_{cr/2}(v)$ and $w_h \notin B_{cr}(v)$. It follows that

$$\text{dist}_G(u_h, h) > cr/2 - (1 + 1/c)r = (c/2 - 1 - 1/c)r.$$  

We already saw that the left-hand side of the above inequality is at most $r$, so this case only arises when $c < \sqrt{6} + 2$. Note also that

$$\text{dist}_G(v, h) \leq \text{dist}_G(v, u_h) + \text{dist}_G(u_h, h) \leq cr/2 + r = (c/2 + 1)r,$$

and hence $B_{c(r/2+1)}(v)$ contains a shortest path $P_h$ from $u_h$ to $h$. Equivalently, $P_h$ is contained in $B_{c\tilde{r}_3}(v)$ for $\tilde{r}_3 = (1/2 + 1/c)r$. Observe that by (2), the length of $P_h$ is greater than

$$(c/2 - 1 - 1/c)r \geq \tilde{r}_3 = (1/2 + 1/c)r,$$

as $c \geq 4$. The length of $P_h$ is of course also bounded by $cr/2$, the maximum length of $Q_h$, and hence $h \in \tilde{W}_3$.

In conclusion, we showed that $W \subseteq \tilde{W}_1 \cup \tilde{W}_2 \cup \tilde{W}_3$, and hence $W$ contains at most $3sk$ elements by Lemma 6.3.

We have now determined all the properties of approximate core hubs that we need in order to prove that any set $X_T$ has low doubling dimension. Recall that for this we need to show that we can cover any ball $B$ of radius $2r$ in the metric defined by $X_T$ by a bounded number of balls of half the radius $r$. We first prove a slightly weaker result in which we show that core hubs in a ball of radius $cr/2$ can be covered by a small number of balls of radius $2r$ for some given $r$. (Note that, for $c > 4$, $2r$ is smaller than
cr/2.) We will later apply the next lemma recursively in order to obtain a bound on the doubling dimension of $X_T$.

**Lemma 6.4.** For any level $i$ and any ball $B_{cr_i/2}(v) \subseteq V$ of radius $cr_i/2$ we can cover $B_{cr_i/2}(v) \cap X_T$ with at most $O(ks \log(1/\varepsilon)/\lambda)$ balls in $V$ of radius $2r_i$ each for any $0 < \varepsilon \leq 2$ and violation $\lambda > 0$.

**Proof.** Recall that $X_T = \bigcup_{i=1}^{l-1} X_T^i$, where $j$ is the level of the town $T$ and $X_T^i$ are the approximate core hubs at level $l$ of $T$. We distinguish three cases based on the level $l$. First consider the vertices in $\bigcup_{i=1}^{l-1} X_T^i$ up to level $i$, and recall that $X_T^i \subseteq \bigcup_{q=1}^{i} (C_q \cap spc(r_i))$, i.e., the approximate core hubs of level $i$ are core hubs of levels up to $i$. By Definition 4.1 the cores of town $T$ form a chain—$C_{q-1} \subseteq C_q$—and thus every vertex of $\bigcup_{q=1}^{i} X_T^q$ is contained in the core $C_i$ of $T$ on level $i$. The core $C_i$ is part of the sprawl of level $i$, which by Definition 3.1 is covered by balls of radius $2r_i$ centered at hubs in $spc(r_i)$. For such a ball to cover some parts of the core $C_i$ in $B_{cr_i/2}(v)$, its center $v$ must be at distance at most $2r_i$ from $B_{cr_i/2}(v)$. Hence by Lemma 6.2 there are at most $3ks$ balls of radius $2r_i$ covering all of $\bigcup_{q=1}^{i} X_T^q$ in $B_{cr_i/2}(v)$.

Second, consider the approximate core hubs on levels $q \in \{i + 1, \ldots, l\}$, where $l = i + \lceil \log_{c/4}(c/\varepsilon) \rceil$. Cover every vertex of $\bigcup_{q=i+1}^{l} X_T^q$ in $B_{cr_i/2}(v)$ by one ball of radius $2r_i$ each. For any such level $q > i$ the radius of $B_{cr_i/2}(v)$ is at most $cr_q/2$. Since we assumed that $\varepsilon \leq 2$ while $c > 4$, the approximate core hubs on level $q$ are shifted at most $cr_q < 2r_i < cr_q/2$ to lower level core hubs by Algorithm 2. Hence we can bound the number of such hubs in $B_{cr_i/2}(v)$ per level by $3ks$ using Lemma 6.2, which also bounds the number of balls we use to cover them. If the violation $\lambda$ tends to zero, the number of such levels is $O(\log_{c/4}(c/\varepsilon)) = O(\log(1/\varepsilon)/\lambda)$, since $\log(c/4) = \log(1 + \lambda/4) = \Theta(\lambda)$. In total this makes $O(ks \log(1/\varepsilon)/\lambda)$ balls for levels up to $l$.

For the remaining levels $l > i + \lceil \log_{c/4}(c/\varepsilon) \rceil$ we use the fact that the approximate core hubs are locally nested by Lemma 6.1. In particular, note that $cr_i \geq cr_l$ since $r_i = (c/4)^i$, i.e., the diameter of $B_{cr_i/2}(v)$ is at most $\varepsilon r_i$ for level $l$. Let $q$ be the lowest level for which $X_T^q \cap B_{cr_i/2}(v) \neq \emptyset$. If $q \leq l$, the hubs in $X_T^q \cap B_{cr_i/2}(v)$ are already accounted for. Otherwise, as before we greedily cover each hub in $X_T^q \cap B_{cr_i/2}(v)$ by a ball of radius $2r_i$, each, and by Lemma 6.2 we need at most $3ks$ balls to do so. Now, by Lemma 6.1, every vertex of $X_T^q \cap B_{cr_i/2}(v)$ for a level $p > \max\{l, q\}$ is contained in some set $X_T^{p'} \cap B_{cr_i/2}(v)$ for $p' \leq \max\{l, q\}$. Since we already covered each hub in $X_T^{p'} \cap B_{cr_i/2}(v)$ with a ball, the claim follows.

We can now use the above lemma recursively to cover the set $X_T$ in a ball $B_{2r_i}(v)$ with balls of half the radius, as we show next.

**Lemma 6.5.** Let $T \in T$ be a town and let $B_{2r_i}(v) \subseteq V$ be a ball of radius $2r_i$. Then $B_{2r_i}(v) \cap X_T$ can be covered by at most $(ks \log(1/\varepsilon)/\lambda)^{O(1/\lambda)}$ balls in $V$ of radius $r_i$ for any $0 < \varepsilon \leq 2$ and violation $\lambda > 0$.

**Proof.** Let $l$ be the smallest level for which $cr_l/2 \geq 2r_i$. Instead of using $B_{2r_i}(v)$ directly, we will cover the larger set $B_{cr_i/2}(v) \cap X_T$ with balls of radius $cr_{l-1}/4 < r_i$, which we find by recursively covering $B_{cr_i/2}(v)$ with balls of the next lower level.

Since $r_i = (c/4)^i$, a ball $B_{2r_{i+1}}(v)$ has radius $2r_{i+1} = 2cr_{i+1} = cr_i/2$. Hence, by Lemma 6.4, we can cover $X_T \cap B_{2r_{i+1}}(h)$ with $O(ks \log(1/\varepsilon)/\lambda)$ balls of radius $2r_i$, on which we recurse. By the choice of $l$, $r > cr_{l-1}/4$, and since $r_i = (c/4)^i$, the number...
of levels $\beta$ on which we need to recurse is at most
\[
\log_{c/4}(cr_1/2) - \log_{c/4}(cr_{\ell - 1}/4) = 1 + \frac{1}{\log_2(c/4)} = O(1/\lambda).
\]
The total number of balls needed to cover $B_{2r}(v)$ with balls of radius $r$ is then at most
\[
\sum_{i=0}^{\beta - 1} O\left(ks \log(1/\varepsilon)/\lambda\right)^i = (ks \log(1/\varepsilon)/\lambda)^{O(1/\lambda)},
\]
which concludes the proof.

The balls $B_r(h)$ found in Lemma 6.5 are centered at hubs. If all these hubs are part of $X_T$, then we have shown that $X_T$ has bounded doubling dimension. However, if $h \notin X_T$ for some ball center, then we have partly covered $B_{2r}(v) \cap X_T$ with invalid balls that are not centered at points in the metric $X_T$. We already addressed this issue in section 4 by proving Lemma 4.3. Thus we are finally ready to prove the remaining part of Theorem 4.2 by bounding the doubling dimension of $X_T$. Consider a ball $B_{2r}(v) \subseteq V$. According to Lemma 6.5 we can cover $B_{2r}(v) \cap X_T$ using at most $(ks \log(1/\varepsilon)/\lambda)^{O(1/\lambda)}$ balls in $V$ of radius $r$. Recall that the doubling dimension is $\log_2\delta$, where $\delta$ is the number of balls needed. Hence by Lemma 4.3 the doubling dimension of $X_T$ is $O(\log(ks \log(1/\varepsilon))/\lambda)$, as claimed.

7. The treewidth of the embedding. We prove by induction that the embedding has bounded treewidth. That is, we prove that the embedding of any town $T \in \mathcal{T}$ has bounded treewidth, assuming that the embeddings of its child towns have bounded treewidth. In particular, we prove the following, which implies the treewidth bound of Theorem 1.3, since there are $O(\log_{c/4}(\alpha)) = O(\log(\alpha)/\lambda)$ levels in total, and we can assume that $s = O(k \log k)$ by [2].

**Theorem 7.1.** The embedding constructed for a town $T \in \mathcal{T}$ of level $j$ has treewidth
\[
j \cdot (\log(\alpha))^{O(\log^2(\frac{\log(\alpha)}{\lambda})/\lambda)}.
\]

To prove Theorem 7.1, we show how to compute a tree decomposition $D_T$ of the embedding $H_T$, when $T$ has child towns in the towns decomposition. Recall that $H_T$ is obtained by connecting the embeddings $H_{T'}$ of each child town $T'$ to the embedding $H_X$ of the approximate core hubs $X_T$. In particular, an edge is added between every vertex in $T'$ and every hub in the connecting bag $b$ of $T'$ in the tree decomposition $D_X$ of $H_X$. To compute $D_T$ we will join the tree decompositions of the child towns with $D_X$. For this we need to inductively specify a root bag for each tree decomposition, and the root bag of $D_T$ is the highest level bag of $D_X$.

Now for each child town $T'$, consider appending the subtree $D_{T'}$ to $D_X$ by adding the root bag of $D_{T'}$ as a child of the connecting bag $b$ of $T'$ in $D_X$. This satisfies condition (a) of Definition 2.1, as the union of all bags is $T$. Unfortunately, though, this initial tree of bags $D_T$ does not satisfy the remaining requirements of a valid tree decomposition of $H_T$ according to Definition 2.1: the edges added to connect the child towns and their connecting bags may not be contained in any bag—violating (b)—and there might be some vertex $v$ for which the bags containing $v$ are not connected in $D_T$—violating (c).

To make $D_T$ valid we change the initial tree of bags in two steps, of which the first will guarantee that (b) is satisfied and the second will guarantee that (c) is satisfied.
Namely, we perform the following for every child town $T'$ and its connecting bag $b$ in $D_X$:

1. add all vertices of $b$ to each bag of $D_{T'}$, and
2. add all hubs of $X_T \cap T'$ to each bag of $D_{T'}$, and also to $b$ and all descendants of $b$ in $D_X$ (but not the descendants of $b$ in $D_T$ that are bags of some $D_{T''}$ for some child town $T'' \neq T'$ of $T$).

We now argue that the resulting tree decomposition is valid.

**Lemma 7.2.** After performing step (1) above, all edges are contained within some bag.

**Proof.** First, note that the decompositions $D_X$ and $D_{T'}$ for each child town $T'$ are valid by Theorem 5.5 and by induction, respectively. Hence the only edges that are not contained in any bag of $D_{T'}$ are those added to connect a child town $T'$ and its connecting bag $b$. We add all vertices of $b$ to every bag of the decomposition $D_{T'}$, so after repeating this for every child town, for every edge in $E(H_T)$ there is a bag in $D_{T'}$ containing both endpoints.

We will bound the growth of the bags during this step later on using the bound on the size of each bag $b$ of $D_X$ given by Theorem 5.5. Next we show that performing the second step will guarantee that (c) of Definition 2.1 is satisfied.

**Lemma 7.3.** After performing step (2) above, for all vertices $v$, the set of bags containing $v$ form a connected subtree of $D_T$, and $D_T$ is a valid tree decomposition of $T$.

**Proof.** Suppose there is a vertex $v$ such that the bags containing $v$ are not connected after performing the first step. By Theorem 5.5 and by induction, the sets of bags containing each vertex are connected within $D_X$ and $D_{T'}$ for all child towns $T'$, so $v$ must be in $X_T \cap T'$ for some $T'$. This means that $v$ is an approximate core hub of $T$ that happens to lie in the child town $T'$. Since child towns of $T$ are disjoint by Lemma 3.3, $v$ cannot be contained in two different ones, so that $T'$ is the only child town containing $v$. Note that $v$ cannot be in the connecting bag $b$ of $T'$ because then the first step would have added $v$ to all bags of $D_{T'}$, which would have connected the sets of bags in $D_X$ and $D_{T'}$ containing $v$. Hence it can only be that $v$ is in a bag of $D_{T'}$ and in some bag of $D_X$ other than the connecting bag of $T'$.

We know from (ii) in Theorem 5.5 that the vertices in the bags of the decomposition $D_{T'}$ for the representative hubs $Y_T$ of $T$ form a hierarchy: every vertex in a bag $b'$ of $D_{T'}$ is also contained in one of the child bags of $b'$. Recall that the decomposition $D_X$ of $X_T$ is obtained from $D_{T'}$ by simply replacing each vertex with all hubs it represents. Hence the vertices in the bags of $D_X$ also form a hierarchy. Furthermore, all hubs in $X_T \cap T'$ are in the same bags in $D_X$, since they are represented by the same vertex of $Y_T$. Since $v \in X_T \cap T'$ is not yet in the connecting bag $b$ of $T'$, this means that in $D_X$ none of the hubs in $X_T \cap T'$ are in a bag on a higher level than $b$.

Recall that we choose the connecting bag $b$ so that its corresponding cluster contains the closest approximate core hub $h$ to $T'$. In this case, $X_T \cap T' \neq \emptyset$ as it contains $v$, so $h$ is a hub in $X_T \cap T'$. By the construction of $D_X$, if $b$ contains $h$, then $b$ contains the entire set $X_T \cap T'$. By (2) of Lemma 5.4, on each level the clusters for $Y_T$ partition $Y_T$. Clearly this is also true for $X_T$. Hence any hub of $X_T \cap T'$, including the problematic vertex $v$, can only be contained in bags of the decomposition $D_X$ that are descendants of $b$.

Due to these observations we add all hubs of $X_T \cap T'$ to each bag of $D_{T'}$ and also to $b$ and all descendants of $b$ in $D_X$, and this will ensure there will not be any $v$ for
which the bags containing it are disconnected in the resulting decomposition. Note that we do not need to add these hubs to descendants of \( b \) in \( D_T \) that are bags of some \( D_{T''} \) for some other child town \( T'' \neq T' \).

For the second part of the lemma, note that adding nodes to bags does not break conditions (a) or (b) of Definition 2.1 established in Lemma 7.2, so the resulting tree decomposition is valid.

At this point we have a valid tree decomposition \( D_T \), but we still need to bound the sizes of the resulting bags in \( D_X \) and each \( D_{T'} \). We use the following two lemmas to bound the size of the bags of \( D_X \). In the first we show that for each bag \( b \) of \( D_X \), the number of child towns connecting to \( b \) and containing approximate core hubs is bounded. In the second lemma we prove a bound on the maximum number of approximate core hubs in each child town.

**Lemma 7.4.** Let \( b \) be a bag of the decomposition \( D_X \) of the embedding \( H_X \) for \( X_T \), and let \( d \) be the doubling dimension of \( X_T \). The number of child towns \( T' \) of \( T \) for which \( X_T \cap T' \neq \emptyset \) and for which \( b \) is their connecting bag is \( O((d/\varepsilon)^d) \).

**Proof.** Let \( Y \subseteq Y_T \) be the set containing exactly one representative for each of child town \( T' \) that has \( b \) as its connecting bag and for which \( X_T \cap T' \neq \emptyset \). We can bound the size of \( Y \) in order to bound the desired number of child towns. To prove the bound we will use the fundamental property of low doubling dimension metrics given by Lemma 5.6, which says that such metrics have a bounded number of vertices in terms of their aspect ratio. We will use this lemma to bound the size of \( Y \) by deriving a bound on its aspect ratio: since the child towns connect to the same bag \( b \), we are able to obtain an upper bound on the distance between the representatives in \( Y \). We also get a lower bound on the distances from the fact that \( b \) was chosen for a child town according to the minimum distance to any other child town.

More concretely, consider the tree decomposition \( D_Y \) for the representative hubs \( Y_T \). The bag \( b \) was obtained from a bag \( b' \) of \( D_Y \) by replacing each vertex with the represented hubs of \( X_T \). If the level of bag \( b' \) is \( \bar{l} \), then, by (3) of Lemma 5.4, the diameter of the cluster \( C' \) corresponding to \( b' \) is at most \( 2^{l+1} \).

Suppose \( T' \) is a child town that has \( b \) as its connecting bag and for which \( X_T \cap T' \neq \emptyset \). The bag \( b \) was chosen so that the corresponding cluster contains the closest hub \( h \) of \( X_T \). Since \( X_T \cap T' \neq \emptyset \), this means \( h \in X_T \cap T' \). Analogous to the connecting bag \( b \), its cluster \( C \) is obtained from cluster \( C' \) by replacing each vertex with its represented hubs. Hence all of \( X_T \cap T' \) resides in \( C \). Accordingly, the representative for the set \( X_T \cap T' \) of each considered child town \( T' \) is in \( C' \), i.e., \( Y \subseteq C' \).

Bags \( b \) and \( b' \) are at the same level \( \bar{l} \). Recall that we chose this level in the following way: if the closest sibling of a child town \( T'' \) is at a distance in the interval \([r_i, r_{i+1}]\), then the level \( \bar{l} \) of \( b \) is \( \min\{j, \bar{l} + \lceil \log_2(d/\varepsilon) \rceil\} \), where \( j \) is the level of the root of \( D_X \) and \( i = \lceil \log_2 r_i \rceil \). Let \( \bar{l}' = \bar{l} - \lceil \log_2(d/\varepsilon) \rceil \) so that \( \bar{l}' \leq \bar{l} \). Thus the distance from \( T'' \) to any of its siblings is more than \( r_{i+1} \geq 2^{i+1} \geq 2^{\bar{l}'-1} \geq \varepsilon 2^{\bar{l}'-1}/d \).

Since each vertex of \( Y \) is in a different child town, the distance between any pair of vertices in \( Y \) is more than \( \varepsilon 2^{\bar{l}'-1}/d \), so the aspect ratio of the set \( Y \) is at most \( 2^{l+1}d/(\varepsilon 2^{l-1}) = O(d/\varepsilon) \), due to the bound on the diameter of cluster \( C' \) containing \( Y \). By Lemma 5.6 we then get \(|Y| \leq O((d/\varepsilon)^d)\), and this bound is the same for the number of considered child towns.

Next we prove that the number of approximate core hubs in each child town is bounded. This result will also help in bounding the treewidth of \( H_X \), since it
gives a bound on the number of approximate core hubs that a vertex from \( Y_T \)
represents.

**Lemma 7.5.** For any child town \( T' \) of \( T \), the number of approximate core hubs in the
intersection \( X_T \cap T' \) is \( O(s \log(1/\varepsilon)/\lambda) \).

**Proof.** Suppose that \( T' \) is a town on level \( i \), and recall from section 6 that
\[ X_T^i \subseteq \bigcup_{q=1}^i C_q \cap \text{spc}(r_q). \]
i.e., the approximate core hubs of level \( i \) are core hubs on levels \( i \) or below. By Definition 4.1
no such core hubs exist, and hence \( T' \) also does not contain any approximate core hubs of level at most \( i \).

Let \( l = i + \lceil \log_{d/4}(1/\varepsilon) \rceil \), and consider \( q \in (i, l] \). Once more, since \( T' \) does not
contain core hubs of level at most \( i \), any approximate core hub of level \( q \) must also be a
core hub of level \( l' \in (i, q] \), and hence we focus on bounding the size of \( \text{spc}(r_{l'}) \cap T' \)
for each \( l' \in (i, l] \). Recall that Lemma 3.2 implies that town \( T' \) has diameter at most
\( r_i \leq cr_{l'}/2 \), and therefore \( T' \) is contained in \( B_{cr_{l'}/2}(v) \) for any \( v \in T' \). Definition 1.2
implies that \( |B_{cr_{l'}/2}(v) \cap \text{spc}(r_{l'})| \leq s \), and hence also \( T' \) contains no more than \( s \)
level \( l' \) core hubs. In summary, we have just shown that the set
\[ X = T' \cap \bigcup_{q \leq l} X_T^q \]
has cardinality at most \( s \lceil \log_{d/4}(1/\varepsilon) \rceil \). It remains to consider levels \( q > l \). Yet again
by Lemma 3.2, \( T' \) has diameter at most
\[ r_i = \left( \frac{c}{l} \right)^l \leq \varepsilon \left( \frac{C}{l} \right)^l < \varepsilon r_q. \]
Lemma 6.1 directly implies that any approximate core hub in \( T' \) of level greater than \( l \)
is contained in \( X \) if the latter set is nonempty. So let us assume that \( X = \emptyset \). In this
case we argue as before, and use Definition 1.2 to bound \( |\text{spc}(r_q) \cap T'| \) by \( s \). All in all,
we showed that \( T' \) contains \( O(s \log c_{l'/4}(1/\varepsilon)) \) approximate core hubs.

Using the obtained bounds in the above lemmas, we are now ready to prove that
the treewidth of the embedding \( H_T \) is bounded.

**Proof of Theorem 7.1.** Towns that have no children are singletons, since every
vertex is a town on level 0. Hence for these the claim is trivially true. Otherwise,
by Lemma 3.3, a town has at least two children. For these we need to bound the
resulting bag sizes of the tree decomposition \( D_T \), as described in this section. First we
determine the treewidth of the embedding \( H_X \) for \( X_T \). The decomposition \( D_X \) was
obtained from the decomposition \( D_Y \) for \( Y_T \) by replacing each vertex with the hubs of
\( X_T \) it represents. For each vertex of \( Y_T \) the number of represented hubs is bounded by
Lemma 7.5, while the treewidth of the embedding for \( Y_T \) is bounded by Theorem 5.5.
Thus if the doubling dimension of \( Y_T \) is \( d \), then the treewidth \( t_X \) of \( H_X \) is
\[ t_X \leq (d \log(\alpha)/\varepsilon')^{O(d)} \cdot s \log(1/\varepsilon)/\lambda. \]
In the first step of the transformation to make the tree decomposition \( D_T \) valid,
we add all vertices of a bag \( b \) of \( D_X \) to all bags of the decomposition trees \( D_T \) of child
towns \( T' \) for which \( b \) is the connecting bag. By Lemma 3.3, if \( T \) is a town on level
\( l \)
was later popularized by the book "What is Mathematics?" in 1941 by Courant and
The aim is to find a minimum cost tree in a set of terminals \( R \subseteq V \) given. The
im in finding the shortest route along the capitals of the lower 48 US states. Later milestones in its study include computing the shortest routes through German cities are suggested to a traveling salesman. The problem became
path metric, visiting all vertices of \( G \). This term is dominated by the asymptotic bound on \( t_X \).
vertices of \( b \). Note that this does not affect the bags of a decomposition \( D_{T''} \) of any child town \( T'' \) of \( T \). By Lemma 7.4, each bag \( b \) of \( D_X \) receives approximate core hubs from \( O((d/\epsilon)\alpha) \) child towns for which \( b \) is the connecting bag. Each such child town adds \( O(s\log(1/\epsilon)\lambda) \) hubs to \( b \) by Lemma 7.5. Hence the total number of hubs added to \( b \) from child towns having \( b \) as their connecting bag is \( O((d/\epsilon)^d\cdot s\log(1/\epsilon)\lambda) \).
However these hubs are also added to all descendants of such a bag \( b \). The total number of levels of the decomposition tree \( D_X \) is \( O(\log(\alpha)) \) by (1) of Lemma 5.4. Hence any bag of \( D_X \) receives at most \( O((d/\epsilon)^d\log(\alpha)\cdot s\log(1/\epsilon)\lambda) \) additional hubs from all its ancestors. This term is again dominated by the asymptotic bound on \( t_X \), since \( \epsilon' = \epsilon^2 \).

It follows that the treewidth of \( D_T \) is \( j \cdot O(t_X) \). Hence to conclude the proof we only need to bound \( t_X \). The doubling dimension \( d \) of \( Y_T \subseteq X_T \) is \( O(\log(\frac{\log(1/\epsilon)}{\lambda})) \) by Theorem 4.2. Since \( x \cdot \log(x)^{O(\log x)} \subseteq \log(x)^{O(x)} \subseteq x^{O(1)} \), and \( O(\log x) \subseteq O(x) \), the treewidth \( t_X \) of \( H_X \) is at most \( \log(\alpha)^{O(\log^2(\frac{x}{\lambda}))}/\lambda \).

8. Obtaining approximation schemes. In this section we demonstrate how we can use the embedding of Theorem 1.3 to derive QPTASs for various network design problems when the input graph \( G = (V,E) \) is an edge-weighted graph with low highway dimension. Specifically, we consider the Travelling Salesman, Steiner Tree, and Facility Location problems. We begin by defining these (see also [39]), and we briefly mention how these problems historically arose in contexts given by transportation networks.

For the Travelling Salesman problem the shortest tour, i.e., cycle in the shortest-path metric, visiting all vertices of \( G \) needs to be found. One of the earliest references\footnote{For historical references see Schrijver [36] and Cook [24].} to the Travelling Salesman problem appears in a manual of 1832, in which five tours through German cities are suggested to a traveling salesman. The problem became known as the “48 States Problem of Hassler Whitney” in 1934 after Whitney studied it in the context of finding the shortest route along the capitals of the lower 48 US states. Later milestones in its study include computing the shortest routes through an increasing number of cities in countries such as the USA, Germany, and Sweden (though these instances used Euclidean distances).

In the Steiner Tree problem, in addition to \( G \) a set of terminals \( R \subseteq V \) is given. The aim is to find a minimum cost tree in \( G \) spanning all terminals (a so-called Steiner tree). An early reference\footnote{For historical references see Brazil et al. [19].} to the Steiner Tree problem appears in a letter by Gauss from 1836, who mentioned it in the context of connecting cities by railways. The problem was later popularized by the book “What is Mathematics?” in 1941 by Courant and Robbins, who described it in terms of minimizing the total length of a road network.

The Facility Location problem assumes additional weights on the vertices, and the goal is to select a subset of vertices \( W \subseteq V \) (the facilities). The opening cost of a facility is given by its vertex weight, and the connection cost of a vertex \( v \in V \) is the distance from \( v \) to the closest facility in \( W \). The objective is to minimize the sum of
all opening and connection costs. The Facility Location problem has the same root\(^3\) as the Steiner Tree problem in the Fermat–Torricelli problem from 1643, in which a point is to be found that minimizes the total distance to three other points in the plane. The generalization to an arbitrary number of other points became known as the Weber problem, after Alfred Weber studied it in 1909 in the context of finding a factory location to minimize the transportation costs of suppliers. Among other problems, Hakimi introduced Facility Location to networks in 1964, and related it to finding locations for police stations in road networks.

The main result of this section is the following, of which we give a proof sketch below.

**Theorem 8.1.** If the input graph \( G \) has constant highway dimension \( k \) with constant violation \( \lambda > 0 \), then for any constant \( \varepsilon \in (0, 1] \) a \( (1 + \varepsilon) \)-approximation to each of the Travelling Salesman, Steiner Tree, and Facility Location problems can be found in quasi-polynomial time.

Our approach is similar to those used for Euclidean [10] and low doubling dimension [38] metrics. Accordingly it can also be used for other problems, as in [10]. The main idea is to compute a bounded treewidth graph from the input according to Theorem 1.3, and then optimally solve the computed graphs using known algorithms for which the running time can be bounded in terms of the treewidth. However, the treewidth bound of Theorem 1.3 depends on the aspect ratio \( \alpha \). To guarantee quasi-polynomial running times we therefore need to ensure that the aspect ratio of the input used in Theorem 1.3 is not too large. We achieve this by computing a coarse net of polynomial aspect ratio for the input graph first. It is not too hard to show that only a small distortion of the optimum solution is incurred if the nets are fine enough, and we therefore obtain approximation schemes for the input instances. However, it is not necessarily the case that the nets themselves are shortest-path metrics of low highway dimension graphs, even if they are obtained from graphs of low highway dimension. Hence we need to argue that we can actually achieve the treewidth bound of Theorem 1.3, even though we use the nets as inputs.

We go on to describe how a QPTAS as claimed in Theorem 8.1 can be obtained, if a problem \( P \) has the following properties. Thereafter we will show that they are true for each of our considered problems.

1. An optimum solution for \( P \) can be computed in time \( n^{O(t)} \) for graphs of treewidth \( t \),
2. a constant approximation to \( P \) in \( G \) can be computed in (quasi-)polynomial time,
3. the diameter of the input graph \( G \) can be assumed to be \( O(n \cdot OPT_G) \), where \( OPT_G \) is the cost of an optimum solution in \( G \),
4. an optimum solution in a \( \delta \)-net of the vertices \( V \) of \( G \) has cost at most \( OPT_G + O(n\delta) \),
5. the optimization function of \( P \) is linear in the edge costs, and
6. any solution of \( P \) in a \( \delta \)-net of the vertices \( V \) of \( G \) can be converted to a solution for \( G \) losing at most an additive factor of \( O(n\delta) \).

Assuming that \( \varepsilon \), the highway dimension \( k \), and the violation \( \lambda \) are constant, the treewidth bound of Theorem 1.3 is polylogarithmic in the aspect ratio \( \alpha \). Combining Theorem 1.3 with an algorithm for bounded treewidth graphs having a running time as proclaimed in item 1 thus does not guarantee quasi-polynomial running time yet,

\(^3\)For historical references see Smith, Laporte, and Harper [37].
since $\alpha$ might be large. Hence we will reduce the aspect ratio by precomputing a 
corase set of vertices of the input first. In particular, we greedily compute a $\delta$-net of $V$, 
where $\delta = \varepsilon k/n$ and $\kappa = \Theta(OPT_G)$ is a constant approximation of the cost $OPT_G$
of the optimum solution for the considered problem, which can be obtained according 
to item 2. We assign each vertex in $V$ to the closest point of the $\varepsilon k/n$-net. Note 
that this point is unique if we assume each shortest-path length to be unique. Since 
the minimum distance between any two vertices of the $\varepsilon k/n$-net is $\Omega(\varepsilon \cdot OPT_G/n)$
and at most $O(n \cdot OPT_G)$ according to item 3, the aspect ratio of the net is $O(n^2/\varepsilon)$. 
For such polynomial aspect ratios, the treewidth guaranteed by Theorem 1.3 yields 
quasi-polynomial $2^{O(polylog(n))}$ running times given an algorithm for bounded treewidth 
graphs as in item 1.

Computing an embedding for the metric given by the $\varepsilon k/n$-net is not straightforward 
though, since the net is not necessarily a metric given by the shortest-path 
distances of a low highway dimension graph. We will therefore use the structure of the 
input graph $G$ and impose it on the computed net. More concretely, a town $T$ on level 
i of $G$ induces a town $T'$ of level $i$ of the $\varepsilon k/n$-net, by restricting $T$ to the vertices 
of the net. All properties such as laminarity, separation bounds, and diameter (see section 3) needed for our construction are maintained by these subsets $T'$. However 
the shortest-path covers are not maintained, since the hubs might not be part of the 
$\varepsilon k/n$-net. Instead of a shortest path cover, for every level $i$ we will use a set of shifted 
hubs. For each hub in $SPC(r_1)$ of $G$ this set of shifted hubs contains the vertex of the 
$\varepsilon k/n$-net it was assigned to, which is at distance at most $\varepsilon k/n$.

Note that the towns decomposition of the net is given by the original hubs of the 
input graph $G$, and not the shifted hubs. Consider the embedding that results from 
using the shifted hubs together with the imposed towns decomposition of the $\varepsilon k/n$-net 
as input to the algorithm. Apart from the fact that towns contain only a subset of the vertices, 
the only difference to using $G$ as input to the algorithm is that the approximate 
core hubs $X_T$ of a town $T$ are now shifted by a total of at most $\varepsilon r_i + \varepsilon k/n$ on level $i$ 
from the original positions of the hubs in $G$. By reexamining the proofs of section 5 it 
isp therefore not hard to see that in the embedding of the net the expected shortest-path 
length for any pair $u, v$ is $(1 + O(\varepsilon))(dist_G(u, v) + O(\varepsilon k/n))$, when using these hubs. 
By item 4 the optimum solution in the $\varepsilon k/n$-net has cost at most $OPT_G + \varepsilon k$, and by item 5 the optimization function is linear in the edge costs. Hence by linearity of 
extpectation, the optimum solution in the embedding, computed by the algorithm given by 
item 1, has expected cost at most $(1 + O(\varepsilon))(OPT_G + O(\varepsilon k)) = (1 + O(\varepsilon))OPT_G$. 
This solution still has to be converted into a solution of the input graph $G$, which 
can be done by item 6 with only an $O(\varepsilon k)$ additive overhead. Hence we obtain an 
approximation scheme.

We still need to argue that we obtain the same treewidth bound of Theorem 1.3 
when using shifted hubs. In particular, it might be that the approximate core hubs 
are not locally sparse, due to the additional $\varepsilon k/n$ shift. To argue that local sparsity 
can be maintained, we make the level $j$ for which $\varepsilon k/n \in [r_j, r_{j+1}]$ the lowest level, 
for any level below $j$ we remove all hubs. Note that the resulting set of hubs 
still covers all distances in the $\varepsilon k/n$-net. The total shift of a hub is now at most 
$\varepsilon k/n + \varepsilon r_i \leq r_{j+1} + \varepsilon r_i \leq (c/4 + \varepsilon)r_i$, since we made $j$ the lowest level. If we assume 
that $\varepsilon \leq 1$, then this shift is less then $cr_i/2$. Accordingly, Lemma 6.2 still implies that 
the hubs in $X_T$ are locally $3k$-sparse, as needed. All other proofs are as before and 
thus we obtain the same treewidth bound as in Theorem 1.3.

Thus if all claimed properties for the considered problems are true, then this gives 
us QPTASs for low highway dimension graphs, as claimed in Theorem 8.1. We will go
on to argue that each of the properties can be maintained for Travelling Salesman, Steiner Tree, and Facility Location. For the latter two, in addition to using a net as input instead of $G$, we also need to specify the additional input parameters. In particular for Steiner Tree, in addition to assigning each vertex of $G$ to the closest net point, we also need to shift terminals. More concretely, if a terminal of $R$ is assigned to a vertex $v$ of the net, then we make $v$ a terminal of the net. For Facility Location we need to adapt the opening costs in the net, which we do by setting the cost of a vertex $v$ in the net to the smallest cost of any vertex of $G$ assigned to $v$.

For each of the three problems, the linearity of the optimization function as required by item 5 is obvious from their definitions. For Travelling Salesman and Steiner Tree, Bateni et al. [16] show how to solve these problems in time $n^{O(1)} \cdot t'$, where $t$ is the treewidth of the input instance. For Facility Location, Ageev [6] gives an $O(n^{t+2})$ algorithm. This settles item 1. It is well-known that a 2-approximation for Travelling Salesman can be obtained from the minimum spanning tree (MST), and that for Steiner Tree the MST on the metric induced by the terminals is a 2-approximation (see, e.g., [39]). Mahdian, Ye, and Zhang [33] give a 1.52-approximation algorithm for the Facility Location problem. Hence we obtain an estimate $\kappa = \Theta(OPT_G)$ in each case, so that also item 2 is true.

It is easy to see that for any instance of the Travelling Salesman problem, $OPT_G$ is at least twice the diameter of the graph $G$. For Steiner Tree, observe that the maximum distance between two terminals is at most $OPT_G$. Therefore we can remove Steiner vertices (vertices that are not terminals) which are farther away from any terminal than $\kappa$. Thus the diameter of $G$ is $O(\kappa)$. For Facility Location, consider a subgraph induced by edges of length at most $\kappa$. Note that in an optimal solution, for any vertex the closest facility will be in its connected component in this subgraph. Hence we can solve the problem on each component separately. The diameter of such a component is at most $O(n\kappa)$. Therefore, we can assume that item 3 is true in each case.

The optimum Travelling Salesman tour in the net is at most $OPT_G$, since the net is a subset of $V$. Since the terminals for the Steiner Tree problem are shifted by at most $\delta$ in a $\delta$-net, the optimum solution in the net has cost at most $OPT_G + n\delta$. By setting the vertex weights of the net as described above for the Facility Location problem, taking each facility of the optimum solution in $G$ and shifting it to the vertex of a $\delta$-net it is assigned to will increase only the total connection cost by at most $n\delta$. Hence the optimum solution in the net (with the adapted vertex weights) has cost at most $OPT_G + n\delta$. This shows item 4 for each problem.

Given a solution of a $\delta$-net of a graph $G$ for Travelling Salesman, we obtain a tour for $G$ by making a detour from each vertex $v$ of the net to the vertices of $G$ assigned to $v$. The total overhead of this step is at most $2n\delta$. For Steiner Tree, we obtain a solution for $G$ by connecting each terminal in $R$ to the terminal of the $\delta$-net it is assigned to. This introduces an additional cost of $n\delta$ in total. The algorithm for Facility Location by Ageev [6] solves a generalization of the problem where the connection cost of each vertex is weighted. More concretely, in addition to the weight determining the opening cost, each vertex $v$ also has a weight $\varphi(v)$, and the connection cost of $v$ for a set $W$ of facilities is $\varphi(v) \cdot \text{dist}(v, W)$. In a $\delta$-net we can set $\varphi(v)$ to be the number of vertices of $G$ assigned to $v$. If a facility is opened on a vertex $v$ of the net, we obtain a solution to $G$ by shifting the facility to the vertex of smallest opening cost assigned to $v$. By our choice of the opening costs in the net, the total opening cost for the solution in $G$ is the same as for the solution in the net. Due to the additional weights $\varphi(v)$, the total connection cost in the solution for $G$ is at most
9. Comparing alternative definitions of the highway dimension. In this section we compare the different definitions of highway dimension, as given in [1, 2, 4] and this paper. We also consider the hardness of computing the highway dimension. The original definition of [1] is the one we consider in the present work (with violation \( \lambda = 0 \) in Definition 1.1). In a follow-up paper [2] a more general definition was given (along with alternative notions such as the average and cardinality-based highway dimension, which we do not consider here). Later in [4] another much more restrictive definition was given, under which graphs of constant highway dimension also have constant doubling dimension. Hence using this definition, the result of Talwar [38] can be applied immediately to obtain a bounded-treewidth embedding with small distortion.

Note that this is not true for graphs of constant highway dimension according to Definition 1.1: a star with unit edge lengths can use the center vertex as the single hub for any scale, since all shortest paths pass through it. Hence its highway dimension is 1, but the doubling dimension of a star is \( \log_2 n \). In the following we will show that in fact a graph that has constant highway dimension according to [4] also has constant highway dimension according to Definition 1.1 if the violation is zero. Hence the original definition of [1] is a generalization of the one used in [4]. As far as we know, this has not been observed anywhere else yet. The highway dimension in [4] is defined as follows.

**Definition 9.1 (see [4]).** Given a shortest path \( P = (v_1, \ldots, v_k) \) and \( r > 0 \), an \( r \)-witness path \( P' \) is a shortest path with length more than \( r \), such that \( P' \) can be obtained from \( P \) by adding at most one vertex to each end. That is, either \( P' = P \), or \( P' = (v_0, v_1, \ldots, v_k) \), or \( P' = (v_1, \ldots, v_k, v_{k+1}) \), or \( P' = (v_0, v_1, \ldots, v_k, v_{k+1}) \). If \( P \) has an \( r \)-witness path \( P' \) it is said to be \( r \)-significant, and \( P \) is \((r, d)\)-close to a vertex \( v \) if \( \operatorname{dist}(P', v) \leq d \). The highway dimension of a graph \( G \) is the smallest integer \( k \) such that for all \( r > 0 \) and \( v \in V \), there is a hitting set of size at most \( k \) for the \( r \)-significant paths that are \((r, 2r)\)-close to \( v \).

The following lemma from [4] implies that an embedding for a graph of constant highway dimension according to Definition 9.1 can easily be obtained by applying Theorem 5.5.

**Lemma 9.2 (see [4]).** A graph that has highway dimension \( k \) according to Definition 9.1 has doubling dimension at most \( \log_2 (k + 1) \).

Lemma 9.2 is also useful to prove that graphs with constant highway dimension according to Definition 9.1 also have constant highway dimension according to Definition 1.1, as we show next.

**Lemma 9.3.** A graph \( G \) that has highway dimension \( k \) according to Definition 9.1 has highway dimension \( O(k^2) \) according to Definition 1.1 for violation \( \lambda = 0 \).

**Proof.** Consider any ball \( B \) of radius \( 4r \) around a vertex \( v \) of \( G \). We need to show that there is a hitting set of size \( O(k^2) \) for all shortest paths of length more than \( r \) entirely contained in \( B \). Since the doubling dimension of \( G \) is at most \( \log_2 (k + 1) \) by Lemma 9.2, there are at most \( k + 1 \) balls of radius \( 2r \) that cover all vertices in \( B \). In particular, any shortest path of length more than \( r \) that is contained in \( B \) also intersects some of the \( k + 1 \) balls of radius \( 2r \). That is, each such shortest path has a vertex that is at distance at most \( 2r \) to some center vertex of one of the \( k + 1 \) balls.
Fig. 6. An example, which has highway dimension 2 according to Definition 9.4, and for which Lemma 6.2 is not true due to $B_4(v)$ and vertices $w_i$.

Each of these balls has a hitting set of size at most $k$ for the $r$-significant paths that are $(r, 2r)$-close to its respective center vertex. Since any shortest path of length more than $r$ is its own $r$-witness, the union of all these hitting sets intersects all the shortest paths of length more than $r$ in $B$. Hence there is a hub set of size $k(k+1)$ that hits all necessary shortest paths in $B$.

We now turn to the more general definition of highway dimension given in [2]. Here the idea is that the hubs need only hit shortest paths that pass through a ball of radius $2r$, instead of shortest paths that are contained in a ball of radius $4r$.

Definition 9.4 (see [2]). The highway dimension of a graph $G$ is the smallest integer $k$ such that for every scale $r > 0$, and every ball $B_{2r}(v)$ of radius $2r$, there are at most $k$ vertices of $V$ hitting all shortest paths of length in $(r, 2r]$ and intersecting $B_{2r}(v)$.

It is easy to see that Definition 9.4 is a generalization of Definition 1.1 for violation $\lambda = 0$, since any path of length at most $2r$ that intersects a ball $B_{4r}(v)$ is also entirely contained in the ball $B_{4r}(v)$. Interestingly however, we do not know how to generalize our embedding results to this more general definition. In particular, we can show that Lemma 6.2 does not hold for graphs of constant highway dimension according to Definition 9.4, as the next lemma implies. Hence an alternative method to the one developed in this paper would be needed to find an embedding of low distortion.

Lemma 9.5. For any integer $l$ there exists a graph with highway dimension $k = 2$ according to Definition 9.4 and the following properties. There is a scale $r > 0$ for which there is a ball $B$ of radius $2r$, such that a minimal locally 2-sparse shortest path cover contains $l+1$ hubs, each of which is at distance at most $2r$ from some vertex in $B$.

Proof. Given $l$ we construct a star-like graph $G$ as follows (see Figure 6). It has a center vertex $v$, and for each $i \in \{1, \ldots, l\}$ it has four vertices $u_i, w_i, x_i, y_i$. There is an edge from $v$ to $u_i$ of length $4$, from $u_i$ to $w_i$ of length $2\varepsilon$, from $w_i$ to $x_i$ of length $1$, and from $w_i$ to $y_i$ of length $1 + \varepsilon$, for some suitably small $\varepsilon > 0$.

We first prove that $G$ has highway dimension $k = 2$ according to Definition 9.4. Consider a ball $B_{2r}(v)$ centered at $v$. If $r < 2$, then this ball contains only $v$ and there is nothing to show. If $r \in [2, 2 + \varepsilon)$, then $B_{2r}(v) = \{v, u_1, \ldots, u_l\}$, and it suffices to choose $v$ as the only hub for this ball: any shortest path intersecting the ball and not containing the hub $v$ has length at most $1 + 3\varepsilon$ (e.g., $u_1w_1y_1$), which is shorter than...
r. If \( r \geq 2 + \varepsilon \), then \( w_i \in B_{2r}(v) \) for all \( i \) and the paths \( x_i w_i y_i \) intersect the ball. It still suffices to choose \( v \) as the only hub since a shortest path that does not contain \( v \) has length at most \( 2 + \varepsilon \) (e.g., \( x_1 w_1 y_1 \)), and only paths of length more than \( r \) need to be hit by the hubs. Now consider a ball \( B_{2r}(z_i) \) for some \( z_i \in \{u_i, w_i, x_i, y_i\} \). If \( r < 4 \), then \( B_{2r}(z_i) \subseteq \{v, u_i, w_i, x_i, y_i\} \), and it suffices to choose \( \{v, w_i\} \) as the hub set since any path intersecting the ball passes through one of these vertices. (If, for instance, \( z_i = u_i \) and \( r = 2 \), then this choice is also necessary due to \( x_i w_i y_i \) and \( v u_i \).)

If \( r \geq 4 \), then it suffices to choose only \( v \) as a hub, since any shortest path not using \( v \) has length at most \( 2 + \varepsilon \).

To prove that the claimed shortest path cover exists, consider the scale \( r = 2 \), for which \( \text{spc}(r) = \{v, w_i \mid 1 \leq i \leq l\} \). This shortest path cover is minimal due to the \( x_i w_i y_i \) paths of length \( 2 + \varepsilon > r \), and the \( v u_i \) paths of length \( 4 = 2r \), for each \( i \). It is also locally 2-sparse since the \( B_{2r}(u_i) \) balls contain the maximum number of two hubs of \( \text{spc}(r) \). Now consider the ball \( B := B_{2r}(v) = \{v, u_1, \ldots, u_l\} \). Even though it contains only the hub \( v \), each hub \( w_i \) has a vertex \( u_i \) in \( B \) at distance \( 2\varepsilon \leq 2r \), which proves the claim.

Note that the graph constructed in the above proof does not have constant highway dimension according to Definition 1.1 with violation \( \lambda = 0 \). This is because at scale \( r = 2 \), the ball centered at \( v \) with radius \( 4r \) contains the \( x_i w_i y_i \) paths, each of which needs to be covered by a hub.

Next we observe that introducing a violation to the original definition of [1] is not an entirely innocuous change. In particular there are graphs for which the highway dimension grows significantly when changing the violation only slightly, as the following lemma shows.

**Lemma 9.6.** For any constant \( c > 4 \) there is a graph that, according to Definition 1.1, has highway dimension \( 1 \) with respect to \( c \) and highway dimension \( \Omega(n) \) with respect to any \( c' > c \).

**Proof.** We construct a spider graph as follows. Let \( l \gg 1 \) be a parameter and \( G = (V, E) \), where \( V = \{u, v_1, w_1, \ldots, v_l, w_l\} \), and \( E = \{(u, v_i), (v_i, w_i) \mid 1 \leq i \leq l\} \), and for all \( i \) the lengths of \((u, v_i)\) and \((v_i, w_i)\) are \( c - 1 \) and 1, respectively. If \( r \geq 1 \), then the hub \( u \) covers all paths longer than \( r \) in any ball of radius \( cr \). Consider a ball \( B_{cr}(t) \) for any vertex \( t \), where \( r < 1 \). If \( t = u \), the hub \( u \) covers all paths in \( B_{cr}(t) \) of length \( r, cr \). If \( t \) is \( v_i \) or \( w_i \) for some \( i \), then \( v_i \) covers all requisite paths in \( B_{cr}(t) \) because \( B_{cr}(t) \) cannot contain \( v_j \) or \( w_j \) for \( j \neq i \). Therefore the highway dimension of \( G \) with respect to \( c \) is \( 1 \).

On the other hand, for any \( c' > c \), let \( r = c/c' \) and consider the ball \( B_{c'r}(u) \), which has radius \( c' \cdot c/c' = c \) and covers the entire graph. Any set of hubs that covers paths of length more than \( c/c' < 1 \) must cover all edges \((v_i, w_i)\) and must therefore include \( v_i \) or \( w_i \) for every \( i \). Hence the highway dimension with respect to \( c' \) is at least \( l = (n - 1)/2 \).

Finally, we also show that computing the highway dimension according to Definition 1.1 is NP-hard. It remains open whether this is also true when considering the more restrictive highway dimension definition from [4].

**Theorem 9.7.** Computing the highway dimension according to Definition 1.1 is NP-hard for any violation \( \lambda \geq 0 \), even on graphs with unit edge lengths.

**Proof.** The reduction is from the NP-hard Vertex Cover problem [28]: given a graph \( G = (V, E) \) we need to compute a minimum sized set of vertices \( C \subseteq V \) hitting each edge, i.e., \( v \in C \) or \( u \in C \) for each \( v u \in E \). For the reduction we introduce an
additional vertex \(w\) and connect it with every vertex in \(V\). Then we give each edge of the resulting graph \(G'\) unit length.

A hub set hitting each shortest path of length 1 is exactly a vertex cover for a graph with unit edge lengths. Note that for scale \(r = 1/e\), the ball \(B_r(w)\) contains all vertices of the graph \(G'\). Hence removing \(w\) from the hub set in \(B_r(w)\), which hits all shortest paths of length more than \(r\), yields a vertex cover for \(G\), as \(c \geq 4\). Conversely, adding \(w\) to a vertex cover for \(G\) is a hub set in \(B_r(w)\) hitting all necessary shortest paths. Thus the highway dimension according to Definition 1.1 is \(k + 1\) in the graph \(G'\) if and only if the smallest vertex cover in \(G\) has size \(k\).

10. Conclusions and open problems. Our main result shows that we can find embeddings of low highway dimension graphs into a distribution of bounded treewidth graphs, with arbitrarily small expected distortion. Since the resulting treewidth is polylogarithmic in the aspect ratio, this implies QPTASs for several optimization problems that naturally arise in transportation networks. Hence, even if the network includes links resulting from means of transportation such as airplanes, trains, or buses, our results indicate that these problems are computationally easier than in the general case. It remains open however to determine the complexity of the considered problems on graphs with constant highway dimension. In particular we do not even know whether the problems are NP-hard for these graphs. Also, it remains open whether we really need the more restrictive highway dimension definition as given in Definition 1.1, or whether the more general one in Definition 9.4 suffices to compute an embedding.

As argued in the introduction, even a complete graph can have highway dimension 1, and therefore low highway dimension graphs do not exclude minors. However it is not clear whether the treewidth of such a graph can be bounded in terms of the aspect ratio \(\alpha\). Even though the hardness results in [27] for the \(p\)-center problem on graphs with highway dimension \(k\) exclude treewidth bounds of the form \(O(k \log \alpha)\), it is possible that the treewidth of such a graph is of the form \(O(\log^b \alpha)\). It seems notoriously difficult however to either prove or disprove this.

Another interesting open problem is the possibility of finding an embedding into a class of graphs with a treewidth that is polylogarithmic in \(1/\varepsilon\) but not the aspect ratio. This would imply PTASs for the considered optimization problems. One limiting factor however is that we use the embedding given by Talwar [38] for low doubling dimension graphs in our construction, for which it is unclear how to obtain embeddings with treewidths independent of the aspect ratio. Even though Bartal, Gottlieb, and Krauthgamer [13] improve on the result by Talwar [38] by giving a PTAS for the Travelling Salesman problem, the latter result does not give an embedding.

One alternative path to obtaining approximation algorithms is to find so-called padded decompositions [3]. Whether these exist for low highway dimension graphs is not known. It may also be possible to find reductions from low highway dimension graphs to graphs of bounded treewidth that distort the optimal solutions of the instances by arbitrarily small factors. That is, the reduction would produce a graph on a different vertex set than the input graph, meaning that it is not an embedding. As for planar graphs [5, 16, 18, 31], this would circumvent the issue that better embeddings might not exist (as shown for the planar case [20, 21]). A last option obviously would be to find algorithms that do not use algorithms for bounded treewidth graphs as a backend, and instead solve the problems on the graphs directly, as, for instance, was done for Euclidean metrics [7, 8, 9] and, in the case of the Travelling Salesman problem, also for low doubling metrics [13].

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
REFERENCES


A Survey on Approximation in Parameterized Complexity: Hardness and Algorithms

Andreas Emil Feldmann 1,* , Karthik C. S. 2,* , Euiwoong Lee 3,* and Pasin Manurangsi 4,*

1 Department of Applied Mathematics (KAM), Charles University, 118 00 Prague, Czech Republic
2 Department of Computer Science, Tel Aviv University, Tel Aviv 6997801, Israel
3 Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA
4 Google Research, Mountain View, CA 94043, USA

* Correspondence: feldmann.a.e@gmail.com (A.E.F.); karthik0112358@gmail.com (K.C.S.); euiwoong@cims.nyu.edu (E.L.); pasin@google.com (P.M.)

Received: 1 October 2019; Accepted: 2 June 2020; Published: 19 June 2020

Abstract: Parameterization and approximation are two popular ways of coping with NP-hard problems. More recently, the two have also been combined to derive many interesting results. We survey developments in the area both from the algorithmic and hardness perspectives, with emphasis on new techniques and potential future research directions.

Keywords: parameterized complexity; approximation algorithms; hardness of approximation

1. Introduction

In their seminal papers of the mid 1960s, Cobham [1] and Edmonds [2] independently phrased what is now known as the Cobham–Edmonds thesis. It states that an optimization problem is feasibly solvable if it admits an algorithm with the following two properties:

1. Accuracy: the algorithm should always compute the best possible (optimum) solution.
2. Efficiency: the runtime of the algorithm should be polynomial in the input size $n$.

Shortly after the Cobham–Edmonds thesis was formulated, the development of the theory of NP-hardness and reducibility identified a whole plethora of problems that are seemingly intractable, i.e., for which algorithms with the above two properties do not seem to exist. Even though the reasons for this phenomenon remain elusive up to this day, this has not hindered the development of algorithms for such problems. To obtain an algorithm for an NP-hard problem, at least one of the two properties demanded by the Cobham–Edmonds thesis needs to be relaxed. Ideally, the properties are relaxed as little as possible, in order to stay close to the notion of feasible solvability suggested by the thesis.

A very common approach is to relax the accuracy condition, which means aiming for approximation algorithms [3,4]. The idea here is to use only polynomial time to compute an $\alpha$-approximation, i.e., a solution that is at most a factor $\alpha$ times worse than the optimum solution obtainable for the given input instance. Such an algorithm may also be randomized, i.e., there is either a high probability that the output is an $\alpha$-approximation, or the runtime is polynomial in expectation.

In a different direction, several relaxations of the efficiency condition have also been proposed. Popular among these is the notion of parameterized algorithms [5,6]. Here the input comes together with some parameter $k \in \mathbb{N}$, which describes some property of the input and can be expected to be small in typical applications. The idea is to isolate the seemingly necessary exponential runtime of NP-hard problems to the parameter, while the runtime dependence on the input size $n$ remains polynomial. In particular, the algorithm should compute the optimum solution in $f(k)n^{O(1)}$ time,
for some computable function \( f : \mathbb{N} \to \mathbb{N} \) independent of the input size \( n \). If such an algorithm exists for a problem it is fixed-parameter tractable (FPT), and the algorithm is correspondingly referred to as an FPT algorithm.

Approximation and FPT algorithms have been studied extensively for the past few decades, and this has lead to a rich literature on algorithmic techniques and deep links to other research fields within mathematics. However, in this process the limitations of these approaches have also become apparent. Some NP-hard problems can fairly be considered to be feasibly solvable in the respective regimes, as they admit polynomial-time algorithms with rather small approximation factors, or can be shown to be solvable optimally with only a fairly small exponential runtime overhead due to the parameter. However, many problems can also be shown not to admit any reasonable algorithms in either of these regimes, assuming some standard complexity assumptions. Thus considering only approximation and FPT algorithms, as has been mostly done in the past, we are seemingly stuck in a swamp of problems for which we have substantial evidence that they cannot be feasibly solved.

To find a way out of this dilemma, an obvious possibility is to lift both the accuracy and the efficiency requirements of the Cobham–Edmonds thesis. In this way, we obtain a parameterized \( \alpha \)-approximation algorithm, which computes an \( \alpha \)-approximation in \( f(k)n^{O(1)} \) time for some computable function \( f \), given an input of size \( n \) with parameter \( k \). The study of such algorithms had been suggested dating back to the early days of parameterized complexity (cf. [5,7–9]), and we refer the readers to an excellent survey of Marx [8] for discussions on the earlier results in the area.

Recently this approach has received some increased interest, with many new results obtained in the past few years, both in terms of algorithms and hardness of approximation. The aim of this survey is to give an overview on some of these newer results. We would like to caution the readers that the goal of this survey is not to compile all known results in the field but rather to give examples that demonstrate the flavor of questions studied, techniques used to obtain them, and some potential future research directions. Finally, we remark that on the broader theme of approximation in \( P \), there was an excellent survey recently made available by Rubinstein and Williams [10] focusing on the approximability of popular problems in \( P \) which admit simple quadratic/cubic algorithms.

Organization of the Survey

The main body of the survey is organized into two sections: one on FPT hardness of approximation (Section 3) and the other on FPT approximation algorithms (Section 4). Before these two main sections, we list some notations and preliminaries in Section 2. Finally, in Section 5, we highlight some open questions and future directions (although a large list of open problems have also been detailed throughout Sections 3 and 4).

2. Preliminaries

In this section, we review several notions that will appear regularly throughout the survey. However, we do not include definitions of basic concepts such as W-hardness, para-NP-hardness, APX-hardness, and so forth; the interested reader may refer to [3–6] for these definitions.

Parameterized approximation algorithms. We briefly specify the different types of algorithms we will consider. As already defined in the introduction, an FPT algorithm computes the optimum solution in \( f(k)n^{O(1)} \) time for some parameter \( k \) and computable function \( f : \mathbb{N} \to \mathbb{N} \) on inputs of size \( n \). The common choices of parameters are the standard parameters based on solution size, structural parameters, guarantee parameters, and dual parameters.

An algorithm that computes the optimum solution in \( f(k)n^{g(k)} \) time for some parameter \( k \) and computable functions \( f,g : \mathbb{N} \to \mathbb{N} \), is called a slice-wise polynomial (XP) algorithm. If the parameter is the approximation factor, i.e., the algorithm computes a \((1 + \epsilon)\)-approximation in \( f(\epsilon)n^{g(\epsilon)} \) time, then it is called a polynomial-time approximation scheme (PTAS). The latter type of algorithm has been studied avant la lettre for quite a while. This is also true for the corresponding FPT algorithm, which
computes a \((1 + \epsilon)\)-approximation in \(f(\epsilon)n^{O(1)}\) time, and is referred to as an efficient polynomial-time approximation scheme (EPTAS). Note that if the standard parameterization of an optimization problem is \(W[1]\)-hard, then the optimization problem does not have an EPTAS (unless \(\text{FPT} = \text{W}[1]\)) \[11\].

Some interesting links between these algorithms, traditionally studied from the perspective of polynomial-time approximation algorithms, and parameterized complexity have been uncovered more recently \[8,11–16\].

As also mentioned in the introduction, a parameterized \(\alpha\)-approximation algorithm computes an \(\alpha\)-approximation in \(f(k)n^{O(1)}\) time for some parameter \(k\) on inputs of size \(n\). If \(\alpha\) can be set to \(1 + \epsilon\) for any \(\epsilon > 0\) and the runtime is \(f(k, \epsilon)n^{\mathcal{O}(\epsilon)}\), then we obtain a parameterized approximation scheme (PAS) for parameter \(k\). Note that this runtime is only truly FPT if we assume that \(\epsilon\) is constant. If we forbid this and consider \(\epsilon\) as a parameter as well, i.e., the runtime should be of the form \(f(k, \epsilon)n^{O(1)}\), then we obtain EPAS.

**Kernelization.** A further topic closely related to the FPT algorithms is kernelization. Here, the idea is that an instance is efficiently pre-processed by removing the “easy parts” so that only the NP-hard core of the instance remains. More concretely, a kernelization algorithm takes an instance \(I\) and a parameter \(k\) of some problem and computes a new instance \(I'\) with parameter \(k'\) of the same problem. The runtime of this algorithm is polynomial in the size of the input instance \(I\) and \(k\), while the size of the output \(I'\) and \(k'\) is bounded as a function of the input parameter \(k\). For optimization problems, it should also be the case that any optimum solution to \(I'\) can be converted into an optimum solution of \(I\) in polynomial time. The new instance \(I'\) is called the kernel of \(I\) (for parameter \(k\)). A fundamental result in fixed-parameter tractability is that an (optimization) problem parameterized by \(k\) is FPT if and only if it admits a kernelization algorithm for the same parameter \[17\]. However the size of the guaranteed kernel will in general be exponential (or worse) in the input parameter. Therefore, an interesting question is whether an NP-hard problem admits small kernels of polynomial size. This can be interpreted as meaning that the problem has a very efficient pre-processing algorithm, which can be used prior to solving the kernel. It also gives an additional dimension to the parameterized complexity landscape.

Kernelization has played a fundamental role in the development of FPT algorithms, where a pre-processing step is often used to simplify the structure of the input instance. It is therefore only natural to consider such pre-processing algorithms for parameterized approximation algorithms as well. Lokshtanov et al. \[18\] define an \(\alpha\)-approximate kernelization algorithm, which computes a kernel \(I'\) such that any \(\beta\)-approximation in \(I'\) can be converted into an \(\alpha\beta\)-approximation to the input instance \(I\) in polynomial time. Again, the size of \(I'\) and \(k'\) need to be bounded as a function of the input parameter \(k\), and the algorithm needs to run in polynomial time. The instance \(I'\) is now called an \(\alpha\)-approximate kernel. Analogous to exact kernels, any problem has a parameterized \(\alpha\)-approximation algorithm if and only if it admits an \(\alpha\)-approximate kernel for the same parameter \[18\], which however might be of exponential size in the parameter.

An \(\alpha\)-approximate kernelization algorithm that computes a polynomial-sized kernel, and for which we may set \(\alpha\) to \(1 + \epsilon\) for any \(\epsilon > 0\), is called a polynomial-sized approximate kernelization scheme (PSAKS). In this case \(\epsilon\) is necessarily considered to be a constant, since any kernelization algorithms needs to run in polynomial time.

We remark here that apart from \(\alpha\)-approximate kernels, there is another common workaround for problems with no polynomial kernels, captured using the notion of Turing kernels. There is also a lower bound framework for Turing kernels \[19\], and the question of approximate kernels for problems that do not even admit Turing kernels is fairly natural to ask. However we skip this discussion for the sake of brevity.

Finally, note that in literature, there is another notion of approximate kernels called \(\alpha\)-fidelity kernelization \[20\] which is different from the one mentioned above. Essentially, an \(\alpha\)-fidelity kernel is a polynomial time preprocessing procedure such that an optimal solution to the reduced instance translates to an \(\alpha\)-approximate solution to the original. This definition allows a loss of precision in the
preprocessing step, but demands that the reduced instance has to be solved to optimality. See [18] for a
detailed discussion on the differences between the two approximate kernel notions.

**Complexity-Theoretic Hypotheses.** We assume that the readers have basic knowledge of (classic)
parameterized complexity theory, including the W-hierarchy, the exponential time hypothesis (ETH),
and the strong exponential time hypothesis (SETH). The reader may choose to recapitulate these
definitions by referring to [6] (Sections 13 and 14).

We will additionally discuss two hypotheses that may not be standard to the community. The first
is the Gap Exponential Time Hypothesis (Gap-ETH), which is a strengthening of ETH. Roughly
speaking, it states that even the approximate version of 3SAT cannot be solved in subexponential
time; a more formal statement of Gap-ETH can be found in Hypothesis 2. Another hypothesis we will
discuss is the Parameterized Inapproximability Hypothesis (PIH), which states that the multicolored
version of the DENSEST $k$-SUBGRAPH is hard to approximate in FPT time. Once again, we do not define
PIH formally here; please refer to Hypothesis 1 for a formal statement.

3. FPT Hardness of Approximation

In this section, we focus on showing barriers against obtaining good parameterized approximation
algorithms. The analogous field of study in the non-parameterized (NP-hardness) regime is the theory
of hardness of approximation. The celebrated PCP Theorem [21,22] and numerous subsequent works
have developed a rich set of tools that allowed researchers to show tight inapproximability results
for many fundamental problems. In the context of parameterized approximation, the field is still in
the nascent stage. Nonetheless, there have been quite a few tools that have already been developed,
which are discussed in the subsequent subsections.

We divide this section into two parts. In Section 3.1, we discuss the results and techniques in
the area of hardness of parameterized approximation under the standard assumption of $W[1] \neq FPT$.
In Section 3.2, we discuss results and techniques in hardness of parameterized approximation under
less standard assumptions such as the Gap Exponential Time Hypothesis, where the gap is inherent in
the assumption, and the challenge is to construct gap-preserving reductions.

3.1. W[1]-Hardness of Gap Problems

In this subsection, we discuss $W[1]$-hardness of approximation of a few fundamental problems.
In particular, we discuss the parameterized inapproximability (i.e., $W[1]$-hard to even approximate)
of the DOMINATING SET problem, the (ONE-SIDED) BICLIQUE problem, the EVEN SET problem,
the SHORTEST VECTOR problem, and the STEINER ORIENTATION problem. We emphasize here that
the main difficulty that is addressed in this subsection is gap generation, i.e., we focus on how to start
from a hard problem (with no gap), say $k$-CLIQUE (which is the canonical $W[1]$-complete problem),
and reduce it to one of the aforementioned problems, while generating a non-trivial gap in the process.

3.1.1. Parameterized Intractability of Biclique and Applications to Parameterized Inapproximability

In this subsubsection, we will discuss the parameterized inapproximability of the one-sided biclique
problem, and show how both that result and its proof technique lead to more inapproximability results.

We begin our discussion by formally stating the $k$-BICLIQUE problem where we are given as input
a graph $G$ and an integer $k$, and the goal is to determine whether $G$ contains a complete bipartite
subgraph with $k$ vertices on each side. The complexity of $k$-BICLIQUE was a long standing open
problem and was resolved only recently by Lin [23] where he showed that it is $W[1]$-hard. In fact,
he showed a much stronger result and this shall be the focus of attention in this subsubsection.
Theorem 1 ([23]). Given a bipartite graph \( G(L \cup R, E) \) and \( k \in \mathbb{N} \) as input, it is \( W[1] \)-hard to distinguish between the following two cases:

- **Completeness:** There are \( k \) vertices in \( L \) with at least \( n^{\Theta(1/\sqrt{k})} \) common neighbors in \( R \);
- **Soundness:** Any \( k \) vertices in \( L \) have at most \( (k + 6)! \) common neighbors in \( R \).

We shall refer to the gap problem in the above theorem as the **ONE-SIDED** \( k \)-**BICLIQUE** problem. To prove the above result, Lin introduced a technique which we shall refer to as Gadget Composition. The gadget composition technique has found more applications since [23]. We provide below a failed approach (given in [23]) to prove the above theorem; nonetheless it gives us good insight into how the gadget composition technique works.

Suppose we can construct a set family \( T = \{S_1, S_2, \ldots, S_n\} \) of subsets of \([n]\) for some integers \( k, n \) and \( h > \ell \) (for example, \( h = n^{1/k} \) and \( \ell = (k + 1)! \)) such that:

- **Property 1:** Any \( k + 1 \) distinct subsets in \( T \) have intersection size at most \( \ell \);
- **Property 2:** Any \( k \) distinct subsets in \( T \) have intersection size at least \( h \).

Then we can combine \( T \) with an instance of \( k \)-**CLIQUE** to obtain a gap instance of **ONE-SIDED** \( k \)-**BICLIQUE** as follows. Given a graph \( G \) and parameter \( k \) with \( V(G) \subseteq [n] \), we construct our instance of **ONE-SIDED** \( k \)-**BICLIQUE**, say \( H(L \cup R, E(H)) \) by setting \( L := E(G) \) and \( R := [n] \), where for any \((v_i, v_j) \in L \) and \( v \in [n] \), we have that \((v_i, v), (v_j, v) \in E(H)\) if and only if \( v \in S_i \cap S_j \). Let \( s := k(k - 1)/2 \). It is easy to check that if \( G \) has a \( k \)-vertex clique, say \( \{v^*_1, \ldots, v^*_k\} \) is a clique in \( G \), then Property 2 implies that \( |\Delta := \bigcap_{i \in [k]} S^*_i| \geq h \). It follows that the set of \( s \) vertices in \( L \) given by \( \{(v^*_i, v^*_j) : \text{for all } \{i, j\} \in \binom{[k]}{2}\} \) are neighbors of every vertex in \( \Delta \subseteq R \). On the other hand, if \( G \) contains no \( k \)-vertex clique, then any \( s \) distinct vertices in \( L \) (i.e., \( s \) edges in \( G \)) must have at least \( k + 1 \) vertices in \( G \) as their end points. Say \( V' \) was the set of all vertices contained the \( s \) edges. By Property 1, we know that \( |\Delta' := \bigcap_{v \in V'} S_v| \leq \ell \), and thus any \( s \) distinct vertices in \( L \) have at most \( \ell \) common neighbors in \( R \).

It is indeed very surprising that this technique can yield non-trivial inapproximability results, as the gap is essentially produced from the gadget and is oblivious to the input! This also stands in stark contrast to the PCP theorem and hardness of approximation results in NP, where all known results were obtained by global transformations on the input. The key difference between the parameterized and NP worlds is the notion of locality. For example, consider the \( k \)-**CLIQUE** problem, if a graph does not have a clique of size \( k \), then given any \( k \) vertices, a random vertex pair in these \( k \) vertices does not have an edge with probability at least \( 1/k^2 \). It is philosophically possible to compose the input graph with a simple error correcting code to amplify this probability to a constant, as we are allowed to blow up the input size by any function of \( k \). In contrast, when \( k \) is not fixed, like in the NP world, \( k \) is of the same magnitude as the input size, and thus we are only allowed to blow up the input size by \( \text{poly}(n) \) factor. Nonetheless, we have to point out that the gadgets typically needed to make the gadget composition technique work must be extremely rich in combinatorial structure (and are typically constructed from random objects or algebraic objects), and were previously studied extensively in the area of extremal combinatorics.

Returning to the reduction above from \( k \)-**CLIQUE** to **ONE-SIDED** \( k \)-**BICLIQUE**, it turns out that we do not know how to construct the set system \( T \), and hence the reduction does not pan out. Nonetheless Lin constructed a variant of \( T \), where Property 2 was more refined and the reduction from \( k \)-**CLIQUE** to **ONE-SIDED** \( k \)-**BICLIQUE**, went through with slightly more effort.

Before we move on to discussing some applications of Theorem 1 and the gadget composition technique, we remark about known stronger time lower bound for **ONE-SIDED** \( k \)-**BICLIQUE** under stronger running time hypotheses. Lin [23] showed a lower bound of \( n^{\Omega(\sqrt{k})} \) for **ONE-SIDED** \( k \)-**BICLIQUE** assuming ETH. We wonder if this can be further improved.
Open Question 1 (Lower bound of One-Sided k-BICLIQUE under ETH and SETH). Can the running time lower bound on One-Sided k-BICLIQUE be improved to \(n^{\Omega(1)}\) under ETH? Can it be improved to \(n^{k-o(1)}\) under SETH?

We remark that a direction to address the above question was detailed in [24]. While on the topic of the k-BICLIQUE problem, it is worth noting that the lower bound of \(n^{\Omega(1/\sqrt{k})}\) for One-Sided k-BICLIQUE assuming ETH yields a running time lower bound of \(n^{\Omega(\log k / \log \log k)}\) for the k-BICLIQUE problem (due to the soundness parameters in Theorem 1). However, assuming randomized ETH, the running time lower bound for the k-BICLIQUE problem can be improved to \(n^{\Omega(\sqrt{k})}\) [23]. Can this improved running time lower bound be obtained just under (deterministic) ETH? Finally, we remark that we shall discuss about the hardness of approximation of the k-BICLIQUE problem in Section 3.2.3.

Inapproximability of k-DOMINATING SET via Gadget Composition. We shall discuss about the inapproximability of k-DOMINATING SET in detail in the next subsubsection. We would like to simply highlight here how the above framework was used by Chen and Lin [25] and Lin [26] to obtain inapproximability results for k-DOMINATING SET.

In [25], the authors starting from Theorem 1, obtain the W[1]-hardness of approximating k-DOMINATING SET to a factor of almost two. Then they amplify the gap to any constant by using a specialized graph product.

We now turn our attention to a recent result of Lin [26] who provided strong inapproximability result for k-DOMINATING SET (we refer the reader to Section 3.1.2 to obtain the context for this result). Lin’s proof of inapproximability of k-DOMINATING SET is a one-step reduction from an instance of k-SET COVER on a universe of size \(O(\log n)\) (where \(n\) is the number of subsets given in the collection) to an instance of k-SET COVER on a universe of size \(\text{poly}(n)\) with a gap of \(\left(\frac{\log n}{\log \log n}\right)^{1/k}\). Lin then uses this gap-producing self-reduction to provide running time lower bounds (under different time hypotheses) for approximating k-set cover to a factor of \((1-o(1))\cdot \left(\frac{\log n}{\log \log n}\right)^{1/k}\). Recall that k-DOMINATING SET is essentially [27,28] equivalent to k-SET COVER.

Elaborating, Lin designs a gadget by combining the hypercube partition gadget of Feige [29] with a derandomizing combinatorial object called universal set, to obtain a gap gadget, and then combines the gap gadget with the input k-SET COVER instance (on small universe but with no gap) to obtain a gap k-SET COVER instance. This is another success story of the gadget composition technique.

Finally, we remark that Lai [30] recently extended Lin’s inapproximability results for dominating set (using the same proof framework) to rule out constant-depth circuits of size \(f(k)n^{o(\sqrt{k})}\) for any computable function \(f\).

Even Set. A recent success story of Theorem 1 is its application to resolve a long standing open problem called k-MINIMUM DISTANCE PROBLEM (also referred to as k-EVEN SET), where we are given as input a generator matrix \(A \in \mathbb{F}_2^{n \times m}\) of a binary linear code and an integer \(k\), and the goal is to determine whether the code has distance at most \(k\). Recall that the distance of a linear code is \(\min_{\alpha \neq 0 \in \mathbb{F}_2^m} \|Ax\|_0\) where \(\|\cdot\|_0\) denote the 0-norm (aka the Hamming norm).

In [31], the authors showed that k-EVEN SET is W[1]-hard under randomized reductions. The result was obtained by starting from the inapproximability result stated in Theorem 1 followed by a series of intricate reductions. In fact they proved the following stronger inapproximability result.

Theorem 2 ([31]). For any \(\gamma \geq 1\), given input \((A,k) \in \mathbb{F}_2^{n \times m} \times \mathbb{N}\), it is W[1]-hard (under randomized reductions) to distinguish between

- Completeness: Distance of the code generated by \(A\) is at most \(k\), and,
- Soundness: Distance of the code generated by \(A\) is more than \(\gamma \cdot k\).
We emphasize that even to obtain the W[1]-hardness of k-EVEN SET (with no gap), they needed to start from the gap problem given in Theorem 1.

The proof of the above theorem proceeds by first showing FPT hardness of approximation of the non-homogeneous variant of k-MINIMUM DISTANCE PROBLEM called the k-NEAREST CODEWORD PROBLEM. In k-MINIMUM CODEWORD PROBLEM, we are given a target vector \( y \) in \( \mathbb{F}^n \) in addition to \((A, k)\), and the goal is to find whether there is any \( x \) (in \( \mathbb{Z}^m \)) such that the Hamming norm of \( Ax - y \) is at most \( k \). As an intermediate step of the proof of Theorem 2, they showed that k-NEAREST CODEWORD PROBLEM is W[1]-hard to approximate to any constant factor.

An important intermediate problem which was studied by [31] to prove the inapproximability of k-NEAREST CODEWORD PROBLEM, was the k-LINEAR DEPENDENT SET problem where given a set \( A \) of \( n \) vectors over a finite field \( \mathbb{F}_q \) and an integer \( k \), the goal is to decide if there are \( k \) vectors in \( A \) that are linearly dependent. They ruled out constant factor approximation algorithms for this problem running in FPT time. Summarizing, the high level proof overview of Theorem 2 follows by reducing ONE-SIDED k-BICLIQUE to k-LINEAR DEPENDENT SET, which is then reduced to k-NEAREST CODEWORD PROBLEM, followed by a final randomized reduction to k-MINIMUM DISTANCE PROBLEM.

Finally, we note that there is no reason to define k-MINIMUM DISTANCE PROBLEM only for binary code, but can instead be defined over larger fields as well. It turns out that [31] cannot rule out FPT algorithms for k-MINIMUM DISTANCE PROBLEM over \( \mathbb{F}_p \) with \( p > 2 \), when \( p \) is fixed and is not part of the input. Thus we have the open problem.

**Open Question 2.** Is it W[1]-hard to decide k-MINIMUM DISTANCE PROBLEM over \( \mathbb{F}_p \) with \( p > 2 \), when \( p \) is fixed and is not part on the input?

**Shortest Vector Problem.** Theorem 1 (or more precisely the constant inapproximability of k-LINEAR DEPENDENT SET stated above) was also used to resolve the complexity of the parameterized k-SHORTEST VECTOR PROBLEM in lattices, where the input (in the \( \ell_p \) norm) is an integer \( k \in \mathbb{N} \) and a matrix \( A \in \mathbb{Z}^{n \times m} \) representing the basis of a lattice, and we want to determine whether the shortest (non-zero) vector in the lattice has length at most \( k \), i.e., whether \( \min_{\|x\|_p \leq k} \|Ax\|_p \leq k \). Again, \( k \) is the parameter of the problem. It should also be noted here that (as in [32]), we require the basis of the lattice to be integer valued, which is sometimes not enforced in literature (e.g., [33,34]). This is because, if \( A \) is allowed to be any matrix in \( \mathbb{R}^{n \times m} \), then parameterization is meaningless because we can simply scale \( A \) down by a large multiplicative factor.

In [31], the authors showed that k-SHORTEST VECTOR PROBLEM is W[1]-hard under randomized reductions. In fact they proved the following stronger inapproximability result.

**Theorem 3 ([31]).** For any \( p > 1 \), there exists a constant \( \gamma_p > 1 \) such that given input \((A, k) \in \mathbb{Z}^{n \times m} \times \mathbb{N}\), it is W[1]-hard (under randomized reductions) to distinguish between

- Completeness: The \( \ell_p \) norm of the shortest vector of the lattice generated by \( A \) is \( \leq k \), and,
- Soundness: The \( \ell_p \) norm of the shortest vector of the lattice generated by \( A \) is \( > \gamma_p \cdot k \).

Notice that Theorem 2 rules out FPT approximation algorithms with any constant approximation ratio for k-EVEN SET. In contrast, the above result only prove FPT inapproximability with some constant ratio for k-SHORTEST VECTOR PROBLEM in \( \ell_p \) norm for \( p > 1 \). As with k-EVEN SET, even to prove the W[1]-hardness of k-SHORTEST VECTOR PROBLEM (with no gap), they needed to start from the gap problem given in Theorem 1.

The proof of the above theorem proceeds by first showing FPT hardness of approximation of the non-homogeneous variant of k-SHORTEST VECTOR PROBLEM called the k-NEAREST VECTOR PROBLEM. In k-NEAREST VECTOR PROBLEM, we are given a target vector \( y \) in \( \mathbb{Z}^m \) in addition to \((A, k)\), and the goal is to find whether there is any \( x \) (in \( \mathbb{Z}^m \)) such that the \( \ell_p \) norm of \( Ax - y \) is at most \( k \). As an
intermediate step of the proof of Theorem 2, they showed that \textit{k-NEAREST VECTOR PROBLEM} is
\textit{W[1]}-hard to approximate to any constant factor. Summarizing, the high level proof overview of
Theorem 3 follows by reducing \textit{ONE-SIDED k-BICLIQUE} to \textit{k-LINEAR DEPENDENT SET}, which is then
reduced to \textit{k-NEAREST VECTOR PROBLEM}, followed by a final randomized reduction to \textit{k-SHORTEST
VECTOR PROBLEM}.

An immediate open question left open from their work is whether Theorem 3 can be extended to
\textit{k-SHORTEST VECTOR PROBLEM} in the $\ell_1$ norm. In other words,

**Open Question 3** (Approximation of \textit{k-SHORTEST VECTOR PROBLEM} in $\ell_1$ norm). Is \textit{k-SHORTEST
VECTOR PROBLEM} in the $\ell_1$ norm in \textit{FPT}?

3.1.2. Parameterized Inapproximability of Dominating Set

In the \textit{k-DOMINATING SET} problem we are given an integer $k$ and a graph $G$ on $n$ vertices as
input, and the goal is to determine if there is a dominating set of size at most $k$. It was a long standing
open question to design an algorithm which runs in time $T(k) \cdot \text{poly}(n)$ (i.e., \textit{FPT-time}), that would
find a dominating set of size at most $F(k) \cdot k$ whenever the graph $G$ has a dominating set of size $k$,
for any computable functions $T$ and $F$.

The first non-trivial progress on this problem was by Chen and Lin [25] who ruled out the
existence of such algorithms (under $\text{W[1]} \neq \text{FPT}$) for all constant functions $F$ i.e., $F(n) = c$, where $c$
is any universal constant). We discussed their proof technique in the previous subsubsection. A couple
of years later, Karthik C. S. et al. [35] completely settled the question, by ruling out the existence of
such an algorithm (under $\text{W[1]} \neq \text{FPT}$) for any computable function $F$. Thus, \textit{k-DOMINATING SET} was
shown to be totally inapproximable. We elaborate on their proof below.

**Theorem 4** ([35]). Let $F : \mathbb{N} \to \mathbb{N}$ be any computable function. Given an instance $(G,k)$ of \textit{k-DOMINATING
SET} as input, it is \textit{W[1]}-hard to distinguish between the following two cases:

- **Completeness**: $G$ has a dominating set of size $k$.
- **Soundness**: Every dominating set of $G$ is of size at least $F(k) \cdot k$.

The overall proof follows by reducing \textit{k-MULTICOLOR CLIQUE} to the gap \textit{k-DOMINATING SET}
with parameters as given in the theorem statement. In the \textit{k-MULTICOLOR CLIQUE} problem, we are
given an integer $k$ and a graph $G$ on vertex set $V := V_1 \cup V_2 \cup \cdots \cup V_k$ as input, where each $V_i$
is an independent set of cardinality $n$, and the goal is to determine if there is a clique of size $k$ in $G$. Following
a straightforward reduction from the \textit{k-CLIQUE} problem, it is fairly easy to see that \textit{k-MULTICOLOR
CLIQUE} is \textit{W[1]}-hard.

The reduction from \textit{k-MULTICOLOR CLIQUE} to the gap \textit{k-DOMINATING SET} proceeds in two
steps. In the first step we reduce \textit{k-MULTICOLOR CLIQUE} to \textit{k-GAP CSP}. This is the step where we
generate the gap. In the second step, we reduce \textit{k-GAP CSP} to gap \textit{k-DOMINATING SET}. This step is
fairly standard and mimics ideas from Feige’s proof of the \textit{NP-hardness of approximating the MAX
COVERAGE} problem [29].

Before we proceed with the details of the above two steps, let us introduce a small technical tool
from coding theory that we would need. We need codes known in literature as good codes, these are
binary error correcting codes whose rate and relative distances are both constants bounded away
from 0 (see [36] (Appendix E.1.2.5) for definitions). The reader may think of them as follows: for every
$\ell \in \mathbb{N}$, we say that $C_\ell \subseteq \{0,1\}^\ell$ is a good code if (i) $|C_\ell| = 2^{\rho \ell}$, for some universal constant $\rho > 0$, (ii) for
any distinct $c,c' \in C_\ell$ we have that $c$ and $c'$ have different values on at least $\delta \ell$ fraction of coordinates,
for some universal constant $\delta > 0$. An encoding of $C_\ell$ is an injective function $E_{C_\ell} : \{0,1\}^{p\ell} \to C_\ell$.
The encoding is said to be efficient if $E_{C_\ell}(x)$ can be computed in $\text{poly}(\ell)$ time for any $x \in \{0,1\}^{p\ell}$.
Let us fix \( k \in \mathbb{N} \) and \( F : \mathbb{N} \to \mathbb{N} \) as in the theorem statement. We further define
\[
\alpha := 1 - \frac{1}{\left(\binom{k}{2} \cdot F(\binom{k}{2})\right)^{\frac{1}{2}}}.
\]

From \( k \)-MULTICOLOR CLIQUE to \( k \)-GAP CSP. Starting from an instance of \( k \)-MULTICOLOR CLIQUE, say \( G \) on vertex set \( V := V_1 \cup V_2 \cup \cdots \cup V_k \), we write down a set of constraints \( \mathcal{P} \) on a variable set \( X := \{x_{ij} \mid i,j \in [k], i \neq j\} \) as follows. For every \( i,j \in [k] \), such that \( i \neq j \), define \( E_{ij} \) to be the set of all edges in \( G \) whose end points are in \( V_i \) and \( V_j \). An assignment to variable \( x_{ij} \) is an element of \( E_{ij} \), i.e., a pair of vertices, one from \( V_i \) and the other from \( V_j \). Suppose that \( x_{ij} \) was assigned the edge \( \{v_i, v_j\} \), where \( v_i \in V_i \) and \( v_j \in V_j \). Then we define the assignment of \( x_{ij}^{0} \) to be \( v_i \) and the assignment of \( x_{ij}^{1} \) to be \( v_j \). We define \( \mathcal{P} := \{P_1, \ldots, P_k\} \), where the constraint \( P_i \) is defined to be satisfied if the assignment to all of \( x_{1j}^{0}, x_{2j}^{0}, \ldots, x_{i-1,j}^{0}, x_{i+1,j}^{0}, \ldots, x_{kj}^{0} \) are the same. We refer to the problem of determining if there is an assignment to the variables in \( X \) such that all the constraints are satisfied as the \( k \)-CSP problem.

Notice that while this is a natural way to write \( k \)-MULTICOLOR CLIQUE as a CSP, where we have tried to check if all variables having a vertex in common, agree on its assignment, there is no gap yet in the \( k \)-CSP problem. In particular, if there was a clique of size \( k \) in \( G \) then there is an assignment to the variables of \( X \) (by assigning the edges of the clique in \( G \) to the corresponding variable in \( X \)) such that all the constraints in \( \mathcal{P} \) are satisfied; however, if every clique in \( G \) is of size less than \( k \) then there every assignment to the variables of \( X \) may violate only one constraint in \( \mathcal{P} \) (and not more).

In order to amplify the gap, we rewrite the set of constraints \( \mathcal{P} \) in a different way to obtain the set of constraints \( \mathcal{P}' \), on the same variable set \( X \), as follows. Suppose that \( x_{ij} \) was assigned the edge \( \{v_i, v_j\} \), where \( v_i \in V_i \) and \( v_j \in V_j \), then for \( \beta \in \{\log n\} \), we define the assignment of \( x_{ij}^{\beta} \) to be the \( \beta \)th coordinate of \( v_i \). Recall that \( |V_i| = n \) and therefore we can label all vertices in \( V_j \) by vectors in \( \{0,1\}^{\log n} \). We define \( \mathcal{P}' := \{P'_1, \ldots, P'_n\} \), where the constraint \( P'_\beta \) is defined to be satisfied if and only if the following holds for all \( i \in [k] \): the assignment to all of \( x_{1j}^{\beta}, x_{2j}^{\beta}, \ldots, x_{i-1,j}^{\beta}, x_{i+1,j}^{\beta}, \ldots, x_{kj}^{\beta} \) are the same.

Again notice that there is an assignment to the variables of \( X \) such that all the constraints in \( \mathcal{P}' \) are satisfied if and only if the same assignment also satisfies all the constraints in \( \mathcal{P}' \).

However, rewriting \( \mathcal{P} \) as \( \mathcal{P}' \) allows us to simply apply the error correcting code \( C_\ell \) (with parameters \( \rho \) and \( \delta \), and encoding function \( E(C_\ell) \)) to the constraints in \( \mathcal{P}' \), to obtain a gap! In particular, we choose \( \ell \) to be such that \( \rho^\ell = \log n \). Consider a new set of constraints \( \mathcal{P}'' \), on the same variable set \( X \), as follows. For any \( z \in \{0,1\}^{\log n} \) and \( \beta \in \{\ell\} \), we denote by \( E(C_\ell)(z)_\beta \), the \( \beta \)th coordinate of \( E(C_\ell)(z) \). We define \( \mathcal{P}'' := \{P''_1, \ldots, P''_n\} \), where the constraint \( P''_\beta \) is defined to be satisfied if and only if the following holds for all \( i \in [k] \): the assignment to all of \( E(C_\ell)(x_{1j}^{\beta})_\beta, E(C_\ell)(x_{2j}^{\beta})_\beta, \ldots, E(C_\ell)(x_{i-1,j}^{\beta})_\beta, E(C_\ell)(x_{i+1,j}^{\beta})_\beta, \ldots, E(C_\ell)(x_{kj}^{\beta})_\beta \) are the same.

Notice, as before, that there is an assignment to the variables of \( X \) such that all the constraints in \( \mathcal{P}' \) are satisfied if and only if the same assignment also satisfies all the constraints in \( \mathcal{P}'' \). However, for every assignment to \( X \) that violates at least one constraint in \( \mathcal{P}' \), we have that the same assignment violates at least \( \delta \) fraction of the constraints in \( \mathcal{P}'' \). To see this, consider an assignment that violates the constraint \( P_i \) in \( \mathcal{P}' \). This implies that there is some \( i \in [k] \) such that the assignment to all of \( x_{1j}^{\beta}, x_{2j}^{\beta}, \ldots, x_{i-1,j}^{\beta}, x_{i+1,j}^{\beta}, \ldots, x_{kj}^{\beta} \) are not the same. Let us suppose, without loss of generality, that the assignment to \( x_{1j}^{\beta} \) and \( x_{2j}^{\beta} \) are different. In other words, we have that \( x_{1j}^{\beta} \neq x_{2j}^{\beta} \), where we think of \( x_{1j}^{\beta}, x_{2j}^{\beta} \) as \( \log n \) bit vectors. Let \( \Delta \subseteq \{\ell\} \) such that \( \beta \in \Delta \) if and only if \( E(C_\ell)(x_{1j}^{\beta})_\beta \neq E(C_\ell)(x_{2j}^{\beta})_\beta \). By the distance of the code \( C_\ell \) we have that \( |\Delta| \geq \delta \ell \). Finally, notice that for all \( \beta \in \Delta \), we have that the assignment does not satisfy constraint \( P_i \) in \( \mathcal{P}'' \). We refer to the problem of distinguishing if there is an assignment to \( X \) such that all the constraints are satisfied or if every assignment to \( X \) does not satisfy a constant fraction of the constraints, as the \( k \)-GAP CSP problem.

In order to rule out \( F(k) \) approximation FPT algorithms for \( k \)-DOMinating SET, we will need that for every assignment to \( X \) that violates at least one constraint in \( \mathcal{P}' \), we have that the same assignment
violates at least $\alpha$ fraction of the constraints in $P''$ (instead of just $\delta$; note that $\alpha$ is very close to 1, whereas $\delta$ can be at most half). To boost the gap [37] we apply a simple repetition/direct-product trick to our constraint system. Starting from $P''$, we construct a new set of constraints $P^*$, on the same variable set $X$, as follows.

$$\mathcal{P}^* := \{ P_S \mid S \in [\ell]^t \},$$

where $t = \frac{\log(1-\alpha)}{\log(1-\delta)}$. For every $S \in [\ell]^t$, we define $P_S$ to be satisfied if and only if for all $\beta \in S$, the constraint $P_\beta$ is satisfied.

It is easy to see that $P''$ and $P^*$ have the same set of completely satisfying assignments. However, for every assignment to $X$ that violates $\delta$ fraction of constraints in $P''$, we have that the same assignment violates at least $\alpha$ fraction of the constraints in $P^*$. To see this, consider an assignment that violates $\delta$ fraction of constraints in $P''$, say it violates all constraints $P_\beta \in P''$, for every $\beta \in \Delta \subseteq [\ell]$. This implies that the assignment satisfies constraint $P_S$ if and only if $S \in ([\ell] \setminus \Delta)^t$. This implies that the fraction of constraints in $P^*$ that the assignment can satisfy is upper bounded by $(1 - \delta)^t = 1 - \alpha$.

**From $k$-Gap CSP to gap $k$-DOMINATING SET.** In the second part, starting from the aforementioned instance of $k$-Gap CSP (after boosting the gap), we construct an instance $H$ of $k$-DOMINATING SET. The construction is due to Feige [29,38] and it proceeds as follows. Let $\mathcal{F}$ be the set of all functions from $\{0,1\}^k$ to $\{0,1\}^t$, i.e., $\mathcal{F} := \{ f : \{0,1\}^k \rightarrow \{0,1\}^t \}$. The graph $H$ is on vertex set $U = A \cup B$, where $A = \mathcal{P}^* \times \mathcal{F}$ and $B = E(G)$, i.e., $B$ is simply the edge set of $G$. We introduce an edge between all pairs of vertices in $B$. We introduce an edge between $a := (S := (s_1, \ldots, s_t) \in [\ell]^t, f : \{0,1\}^k \rightarrow \{0,1\}^t) \in A$ and $e := (v_i, v_j) \in E$ if and only if the following holds.

$$\exists \tau := (\tau_1, \ldots, \tau_t) \in \{0,1\}^{kt}, \text{ such that } f(\tau) = (i, j) \text{ and }$$

$$\forall \tau \in [\ell], \text{ we have } \mathcal{E}_{C_\tau}(v_i) = (\tau_i), \text{ and } \mathcal{E}_{C_\tau}(v_j) = (\tau_j).$$

Notice that the number of vertices in $H$ is $|A| + |B| \leq \left(\frac{\log n}{p}\right)^t \cdot \left(\frac{k}{2}\right)^{2k} + n^2 < \eta(k) \cdot n^{2.01}$, for some computable function $\eta$. It is not hard to check that the following hold:

- (Completeness) If there is an assignment to $X$ that satisfies all constraints in $P^*$, then the corresponding $\left(\frac{k}{2}\right)$ vertices in $B$ dominate all vertices in the graph $H$.
- (Soundness) If each assignment can only satisfy $(1 - \alpha)$ fraction of constraints in $P^*$, then any dominating set of $H$ has size at least $F \left(\frac{k}{2}\right) \cdot \left(\frac{k}{2}\right)$.

We skip presenting details of this part of the proof here. The proofs have been derived many times in literature; if needed, the readers may refer to Appendix A of [35]. This completes our sketch of the proof of Theorem 4.

A few remarks are in order. First, the $k$-Gap CSP problem described in the proof above, is formalized as the $k$-MAXCOVER problem in [35] (and was originally introduced in [39]). In particular, the formalism of $k$-MAXCOVER (which may be thought of as the parameterized label cover problem) is generic enough to be used as an intermediate gap problem to reduce to both $k$-DOMINATING SET (as in [35]) and $k$-CLIQUE (as in [39]). Moreover, it was robust enough to capture stronger running time lower bounds (under stronger hypotheses); this will elaborated below. However, in order to keep the above proof succinct, we skipped introducing the $k$-MAXCOVER problem, and worked with $k$-Gap CSP, which was sufficient for the above proof.

Second, Karthik C. S. et al. [35] additionally showed that for every computable functions $T, F : \mathbb{N} \rightarrow \mathbb{N}$ and every constant $\epsilon > 0$:

- Assuming the Exponential Time Hypothesis (ETH), there is no $F(k)$-approximation algorithm for $k$-DOMINATING SET that runs in $T(k) \cdot n^{o(k)}$ time.
- Assuming the Strong Exponential Time Hypothesis (SETH), for every integer $k \geq 2$, there is no $F(k)$-approximation algorithm for $k$-DOMINATING SET that runs in $T(k) \cdot n^{k-\epsilon}$ time.
In order to establish Theorem 4 and the above two results, Karthik C. S. et al. [35] introduced a framework to prove parameterized hardness of approximation results. In this framework, the objective was to start from either the W[1] ≠ FPT hypothesis, ETH, or SETH, and end up with the gap $k$-DOMINATING SET, i.e., they design reductions from instances of $k$-CLIQUE, 3-CNFSAT, and $\ell$-CNFSAT, to an instance of gap $k$-DOMINATING SET. A prototype reduction in this framework has two modular parts. In the first part, which is specific to the problem they start from, they generate a gap and obtain hardness of gap $k$-MAXCOVER. In the second part, they show a gap preserving reduction from gap $k$-MAXCOVER to gap $k$-DOMINATING SET, which is essentially the same as the reduction from $k$-GAP CSP to $k$-DOMINATING SET in the proof of Theorem 4.

The first part of a prototype reduction from the computational problem underlying a hypothesis of interest to gap $k$-MAXCOVER follows by the design of an appropriate communication protocol. In particular, the computational problem is first reduced to a constraint satisfaction problem (CSP) over $k$ (or some function of $k$) variables over an alphabet of size $n$. The predicate of this CSP would depend on the computational problem underlying the hypothesis from which we started. Generalizing ideas from [40], they then show how a protocol for computing this predicate in the multiparty (number of players is the number of variables of the CSP) communication model, can be combined with the CSP to obtain an instance of gap $k$-MAXCOVER. For example, for the W[1] ≠ FPT hypothesis and ETH, the predicate is a variant of the equality function, and for SETH, the predicate is the well studied disjointness function. The completeness and soundness of the protocols computing these functions translate directly to the completeness and soundness of $k$-MAXCOVER.

Third, we recall that Lin [26] recently provided alternate proofs of Theorem 4 and the above mentioned stronger running time lower bounds. While we discussed about his proof technique in Section 3.1.1, we would like to discuss about his result here. Following the right setting of parameters in the proof of Theorem 4 (for example set $\alpha = 1 - \frac{1}{(\log n)^{O(1)}}$), we can obtain that approximating $k$-DOMINATING SET to a factor of $(\log n)^{1/k^3}$ is $W[1]$-hard. Lin improved the exponent of $1/k^3$ in the approximation factor to $h(k)$ for any computable function $h$. Can this inapproximability be further improved? On the other hand, can we do better than the simple polynomial time greedy algorithm which provides a $(1+\ln n)$ factor approximation? This leads us to the following question:

**Open Question 4** (Tight inapproximability of $k$-DOMINATING SET). Is there a $(\log n)^{1-o(1)}$ factor approximation algorithm for $k$-DOMINATING SET running in time $n^{k^{1+0.1}}$?

We conclude the discussion on $k$-DOMINATING SET with an open question on W[2]-hardness of approximation. As noted earlier, $k$-DOMINATING SET is a W[2]-complete problem, and Theorem 4 shows that the problem is W[1]-hard to approximate to any $F(k)$ factor. However, is there some computable function $F$ for which approximating $k$-DOMINATING SET is in W[1]? In other words we have:

**Open Question 5** (W[2]-completeness of approximating $k$-DOMINATING SET). Can we base total inapproximability of $k$-DOMINATING SET on W[2] ≠ FPT?

### 3.1.3. Parameterized Inapproximability of Steiner Orientation by Gap Amplification

Gap amplification is a widely used technique in the classic literatures on (NP-)hardness of approximation (e.g., [41–43]). In fact, the arguably simplest proof of the PCP theorem, due to Dinur [43], is indeed via repeated gap amplification. The overall idea here is simple: we start with a hardness of approximation for a problem with small factor (e.g., $1 + 1/n$). At each step, we perform an operation that transforms an instance of our problem to another instance, in such a way that the gap becomes bigger; usually this new instance will also be bigger than our instance. By repeatedly applying this operation, one can finally arrive at a constant, or even super constant, factor hardness of approximation.
There are two main parameters that determine the success/failure of such an approach: how large the new instance is compared to the old instance (i.e., size blow-up) and how large the new gap is compared to the old gap, in each operation. To see how these two come into the picture, let us first consider a case study where a (straightforward) gap amplification procedure does not work: \textsc{k-CLIQUE}. The standard way to amplify the gap for \textsc{k-CLIQUE} is through graph product. Recall that the (tensor) graph product of a graph $G = (V, E)$ with itself, denoted by $G^\otimes 2$, is a graph whose vertex set is $V^2$ and there is an edge between $(u_1, u_2)$ and $(v_1, v_2)$ if and only if $(u_1, v_1) \in E$ and $(u_2, v_2) \in E$. It is not hard to check that, if we can find a clique of size $t$ in $G^\otimes 2$, then we can find one of size $\sqrt{t}$ in $G$ (and vice versa). This implies that, if we have an instance of clique that is hard to approximate to within a factor of $(1 + \varepsilon)$, then we may take the graph product with itself which yields an instance of \textsc{CLIQUE} that is hard to approximate to within a factor of $(1 + \varepsilon)^2$.

Now, let us imagine that we start with the hard instance of an exact version of \textsc{k-CLIQUE}. We may think of this as being hard to approximate to within a factor of $(1 - 1/k)$. Hence, we may apply the above gap amplification procedure $\log k$ times, resulting in an instance of \textsc{CLIQUE} that is hard to approximate to within a factor of $(1 - 1/k)^{\log k}$, which is a constant bounded away from one (i.e., $\approx 1/e$). The bad news here is that the number of the vertices of the final graph is $n^{2\log k} = n^k$, where $n$ is the number of vertices of the initial graph. This does not give any lower bound, because we can solve \textsc{k-CLIQUE} in the original graph in $n^{O(k)}$ time trivially! In the next subsection, we will see a simple way to prove hardness of approximating \textsc{k-CLIQUE}, assuming stronger assumptions. However, it remains an interesting and important open question how to prove such hardness from a non-gap assumption:

**Open Question 6.** Is it \textsc{W[1]}-hard or ETH-hard to approximate \textsc{k-CLIQUE} to within a constant factor in \textsc{FPT} time?

Having seen a failed attempt, we may now move on to a success story. Remarkably, Wlodarczyk [44] recently managed to use gap amplification to prove hardness of approximation for connectivity problems, including the \textsc{k-STEINER ORIENTATION} problem. Here we are given a mixed graph $G$, whose edges are either directed or undirected, and a set of $k$ terminal pairs $\{(s_i, t_i)\}_{i \in [k]}$. The goal is to orient all the undirected edges in such a way that maximizes the number of $t_i$ that can be reached from $s_i$. The problem is known to be in \textsc{XP} [45] but is \textsc{W[1]}-hard even when all terminal pairs can be connected [46]. Starting from this \textsc{W[1]}-hardness, Wlodarczyk [44] devises a gap amplification step that implies a hardness of approximation with factor $(\log k)^{O(1)}$ for the problem. Due to the technicality of the gap amplification step, we will not go into the specifics in this survey. However, let us point out the differences between this gap amplification and the (failed) one for \textsc{CLIQUE} above. The key point here is that the new instance of Wlodarczyk’s gap amplification has size of the form $f(k) \cdot n$ instead of $n^2$ as in the graph product. This means that, even if we are applying Wlodarczyk’s gap amplification step $\log(k)$ times, or, more generally, $g(k)$ times, it only results in an instance of size $\frac{f(f(\cdots (f(k))))}{g(k)} \cdot n$, which is still \textsc{FPT}. Since the technique is still quite new, it is an exciting frontier to examine whether other parameterized problems allow such similar gap amplification steps.

### 3.2. Hardness from Gap Hypotheses

In the previous subsection, we have seen that several hardness of approximation results can be proved based on standard assumptions. However, as alluded to briefly, some basic problems, including \textsc{k-CLIQUE}, still evades attempts at proving such results. This motivates several researchers in the community to come up with new assumptions that allow more power and flexibility in proving inapproximability results. We will take a look at two of these hypotheses in this subsection; we note that there have also been other assumptions formulated, but we only focus on these two since they arguably have been used most often.
The first assumption, called the Parameterized Inapproximability Hypothesis (PIH) for short, can be viewed as a gap analogue of the W[1] ̸= FPT assumption. There are many (equivalent) ways to state PIH. We choose to state it in terms of an inapproximability of the colored version of DENSEST k-SUBGRAPH. In MULTICOLORED DENSEST k-SUBGRAPH, we are given a graph $G = (V, E)$ where the vertex set $V$ is partitioned into $k$ parts $V_1, \ldots, V_k$. The goal is to select $k$ vertices $v_1 \in V_1, v_2 \in V_2, \ldots, v_k \in V_k$ such that $\{v_1, \ldots, v_k\}$ induces as many edges as possible.

It is easy to see that the exact version of this problem is W[1]-hard, via a straightforward reduction from $k$-CLIQUE. PIH postulates that even the approximate version of this problem is hard:

**Hypothesis 1** (Parameterized Inapproximability Hypothesis (PIH) [47, 48]). For some constant $\epsilon > 0$, there is no $(1 + \epsilon)$ factor FPT approximation algorithm for MULTICOLORED DENSEST k-SUBGRAPH.

There are two important remarks about PIH. First, the factor $(1 + \epsilon)$ is not important, and the conjecture remains equivalent even if we state it for a factor $C$ for any arbitrarily large constant $C$; this is due to gap amplification via parallel repetition [42]. Second, PIH implies that $k$-CLIQUE is hard to approximate to within any constant factor:

**Lemma 1.** Assuming PIH, there is no constant factor FPT approximation algorithm for $k$-CLIQUE.

The above result can be shown via a classic reduction of Feige, Goldwasser, Lovász, Safra and Szegedy (henceforth FGLSS) [49], which was one of the first works connecting proof systems and hardness of approximation. Specifically, the FGLSS reduction transforms $G$ to another graph $G'$ by viewing the edges of $G$ as vertices of $G'$. Then, we connect $\{u_1, v_1\}$ and $\{u_2, v_2\}$ except when the union $\{u_1, v_1\} \cup \{u_2, v_2\}$ contains two distinct vertices from the same partition. One can argue that the size of the largest clique in $G'$ is exactly equal to the number of edges in the optimal solution of MULTICOLORED DENSEST k-SUBGRAPH on $G$. As a result, PIH implies hardness of approximation of the former. Interestingly, however, it is not known if the inverse is true and this remains an interesting open question:

**Open Question 7.** Does PIH hold if we assume that $k$-CLIQUE is FPT inapproximable to within any constant factor?

As demonstrated by the FGLSS reduction, once we have a gap, it is much easier to give a reduction to another hardness of approximation result, because we do not have to create the initial gap ourselves (as in the previous subsection) but only need to preserve or amplify the gap. Indeed, PIH turns out to be a pretty robust hypothesis that gives FPT inapproximability for many problems, including $k$-CLIQUE, DIRECTED ODD CYCLE TRAVERSAL [48] and STRONGLY CONNECTED STEINER SUBGRAPH [50]. We remark that the current situation here is quite similar to that of the landscape of the classic theory of hardness of approximation before the PCP Theorem [21, 22] was proved. There, Papadimitriou and Yannakakis introduced a complexity class MAX-SNP and show that many optimization problems are hard (or complete) for this class [51]. Later, the PCP Theorem confirms that these problems are NP-hard. In our case of FPT inapproximability, PIH seems to be a good analogy of MAX-SNP for problems in W[1] and, as mentioned before, PIH has been used as a starting point of many hardness of approximation results. However, there has not yet been many reverse reductions to PIH, and this is one of the motivation behind Question 7 above.

Despite the aforementioned applications of PIH, there are still quite a few questions that seem out of reach of PIH, such as whether there is an $o(k)$ factor FPT approximation for $k$-CLIQUE or questions related to running time lower bounds of approximation algorithms. On this front, another stronger conjecture called the Gap Exponential Time Hypothesis (Gap-ETH) is often used instead:
Hypothesis 2 (Gap Exponential Time Hypothesis (Gap-ETH) [52,53]). For some constants $\epsilon, \delta > 0$, there is no $O(2^{kn})$-time algorithm that can, given a 3CNF formula, distinguish between the following two cases:

- (Completeness) the formula is satisfiable.
- (Soundness) any assignment violates more than $\epsilon$ fraction of the clauses.

Here $n$ denotes the number of clauses [54].

Clearly, Gap-ETH is a strengthening of ETH, which can be thought of in the above form but with $\epsilon = 1/n$. Another interesting fact is that Gap-ETH is stronger than PIH. This can be shown via the standard reduction from 3SAT to $k$-CLIQUE that establishes $N^{O(k)}$ lower bound for the latter. The reduction, due to Chen et al. [55,56], proceed as follows. First, we partition the set of clauses $C$ into $C_1, \ldots, C_k$ each of size $n/k$. For each $C_i$, we create a partition $V_i$ in the new graph where each vertex corresponds to all partial assignments (to variables that appear in at least one clause of $C$) that satisfy all the clauses in $C_i$. Two vertices are connected if the corresponding partial assignments are consistent, i.e., they do not assign a variable to different values.

If there is an assignment that satisfies all the clauses, then clearly the restrictions of this assignment to each clause corresponds to $k$ vertices from different partitions that form a clique. On the other hand, it is also not hard to argue that, in the soundness case, the number of edges induced by any $k$ vertices from different partitions is at most $1 - \Theta(\epsilon)$. Thus, Gap-ETH implies PIH as claimed.

Now that we have demonstrated that Gap-ETH is at least as strong as PIH, we may go further and ask how much more can we achieve from Gap-ETH, compared to PIH. The obvious consequences of Gap-ETH is that it can give explicit running time lower bounds for FPT hardness of approximation results. Perhaps more surprising, however, is that it can be used to improve the inapproximability ratio as well. The rest of this subsection is devoted to present some of these examples, together with brief overviews of how the proofs of these results work.

3.2.1. Strong Inapproximability of $k$-Clique

Our first example is the $k$-CLIQUE problem. Obviously, we can approximate $k$-CLIQUE to within a factor of $k$, by just outputting any single vertex. It had long been asked whether an $o(k)$-approximation is achievable in FPT time. As we saw above, PIH implies that a constant factor FPT approximation does not exist, but does not yet resolve this question. Nonetheless, assuming Gap-ETH, this question can be resolved in the negative:

Theorem 5 ([39]). Assuming Gap-ETH, there is no $o(k)$-FPT-approximation for $k$-CLIQUE.

The reduction used in [39] to prove the above inapproximability is just a simple modification of the above reduction [55,56] that we saw for $k$-CLIQUE. Suppose that we would like to rule out a $\frac{k}{g}$-approximation, where $g = g(k)$ is a function such that $\lim_{k \to \infty} g(k) = \infty$. The only change in the reduction is that, instead of letting $C_1, \ldots, C_k$ be the partition of the set of clauses $C$, we let each $C_i$ be a set of $\frac{Dn}{k}$ clauses for some sufficiently large constant $D > 0$. The rest of the reduction works similar to before: for each $C_i$, we create a vertex corresponding to each partial assignment that satisfies all the clauses in $C_i$. Two vertices are joined by an edge if and only if they are consistent. This completes the description of the reduction.

To see that the reduction yields Theorem 5, first note that, if there is an assignment that satisfies the CNF formula, then we can again pick the restrictions on this formula onto $C_1, \ldots, C_k$; these gives $k$ vertices that induces a clique in the graph.

On the other hand, suppose that every assignment violates more than an $\epsilon$ fraction of clauses. We will argue that there is no clique of size $g$ in the constructed graph. The only property we need from the subsets $C_1, \ldots, C_k$ is that the union of any $g$ such subsets contain at least $(1 - \epsilon)$ fraction of the clauses. It is not hard to show that this is true with high probability, when we choose $D$ to be
sufficiently large. Now, suppose for the sake of contradiction that there exists a clique of size \( g \) in the graph. Since the vertices corresponding to the same subset \( C_i \) form an independent set, it must be that these \( g \) vertices are from different subsets. Let us call these subsets \( C_{i_1}, \ldots, C_{i_g} \). Because these vertices induce a clique, we can find a global assignment that is consistent with each vertex. This global assignment satisfies all the clauses in \( C_{i_1} \cup \cdots \cup C_{i_g} \). However, \( C_{i_1} \cup \cdots \cup C_{i_g} \) contains at least \( 1 - \epsilon \) fraction of all clauses, which contradicts to our assumption that every assignment violates more than \( \epsilon \) fraction of the clauses.

Now, if we can \( o(k) \)-approximate \( k \)-CLIQUE in \( T(k) \cdot N^{O(1)} \) time. Then, we may run this algorithm to distinguish the two cases in Gap-ETH in \( f(k) \cdot (2^{\Omega(n)})^{O(1)} = 2^{o(n)} \) time, which violates Gap-ETH. This concludes our proof sketch. We end by remarking that the reduction may also be viewed as an instantiation of the randomized graph product \([41,57,58]\), and it can also be derandomized. We omit the details of the latter here. Interested readers may refer to \([39]\) for more detail.

### 3.2.2. Strong Inapproximability of Multicolored Densest \( k \)-Subgraph and Label Cover

For our second example, we go back to the MULTICOLORED DENSEST \( k \)-SUBGRAPH once again. Recall that PIH asserts that this problem is hard to approximate to some constant factor, and we have seen above that Gap-ETH also implies this. On the approximation front, however, only the trivial \( k \)-approximation algorithm is known: just pick a vertex that has edges to as many partitions as possible. Then, output that vertex and one of its neighbors from each partition. It is hence a natural question to ask whether it is possible to beat this approximation ratio. This question has been, up to lower order terms, answered in the negative, assuming Gap-ETH:

**Theorem 6** ([39]). *Assuming Gap-ETH, there is no \( k^{1-o(1)} \)-approximation for MULTICOLORED DENSEST \( k \)-SUBGRAPH.*

An interesting aspect of the above result is that, even in the NP-hardness regime, no NP-hardness of factor \( k^\gamma \) for some constant \( \gamma > 0 \) is known. In fact, the problem is closely related to (and is a special case of) a well-known conjecture in the hardness of approximation community called the Sliding Scale Conjecture (SSC) \([60–62]\). (See \([59]\) for more discussion on the relation between the two.) Thus, this is yet another instance where taking a parameterized complexity perspective helps us advance knowledge even in the classical settings.

To prove Theorem 6, arguably the most natural reduction here is the above reduction for Clique! Note that we now view the vertices corresponding to each subset \( C_i \) as forming a partition \( V_i \). The argument in the YES case is exactly the same as before: if the formula is satisfiable, then there is a (multicolored) \( k \)-clique. However, as the readers might have noticed, the argument in the NO case does not go through anymore. In particular, even when the graph is quite dense (e.g., having half of the edges present), it may not contain any large clique at all and hence it is unclear how to recover back an assignment that satisfies a large fraction of constraints.

This obstacle was overcome in \([59]\) by proving an agreement testing theorem (i.e., direct product theorem), which is of the following form. Given \( k \) local functions \( f_1, \ldots, f_k \), where \( f_i : S_i \to \{0, 1\} \) is a boolean function whose domain \( S_i \) is a subset of a universe \( U \). If some (small) \( \zeta \) fraction of the pairs agree \([63]\) with each other, then we can find (i.e., “decode”) a global function \( h : [n] \to \{0, 1\} \) that “approximately agrees” with roughly \( \zeta \) fraction of the local functions. The theorem in \([59]\) works when \( S_1, \ldots, S_k \) are sets of size \( \Omega(n) \).

Due to the technical nature of the definitions, we will not fully formalize the notions in the previous paragraph. Nonetheless, let us sketch how to apply the agreement testing theorem to prove the NO case for our reduction. Suppose for the sake of contradiction that the formula is not \((1 - \delta)\)-satisfiable and that there exists a \( k \)-subgraph with density \( \zeta \geq \frac{1}{k^{\frac{1}{\gamma(1-\delta)}}} \). Recall that each selected vertex is simply a partial assignment onto the subset of clauses \( C_i \) for some \( i \); we may view this as a function \( f_i : S_i \to \{0, 1\} \) where \( S_i \) denote the set of variables that appear in \( C_i \). Here the universe \( U \) is
the set of all variables. With this perspective, we can apply the agreement testing theorem to recover
a global function \( h : \mathcal{U} \to \{0, 1\} \) that “approximately agrees” with roughly \( \zeta k \) of the local functions. Notice that, in this context, \( h \) is simply a global assignment for the CNF formula. Previously in the
proof for inapproximability of Clique, we had a global assignment that (perfectly) agrees with \( g \) local functions, from which we can conclude that this assignment satisfies all but \( \delta \) fraction of the clauses. It turns out that relaxing “perfect agreement” to “approximate agreement” does not affect the proof too much, and the latter still implies that \( h \) satisfies all but \( \delta \) fraction of clauses as desired.

As for the proof of the agreement testing theorem itself, we will not delve too much into detail here. However, we note that the proof is based on looking at different “agreement levels” and the graph associated with them. It turns out that such a graph has a certain transitivity property, which allows one to “decode” back the global function \( h \). This general approach of looking at different agreement levels and their transitivity properties is standard in the direct product/agreement testing literature [64–66]. The main challenge in [59] is to make the proof works for \( \zeta \) as small as \( 1/k \), which requires a new notion of transitivity.

To end this subsection, we remark that the MULTICOLORED DENSEST \( k \)-SUBGRAPH is known as the
2-ary Constraint Satisfaction Problem (2-CSP) in the classical hardness of approximation community. The problem, and in particular its special case called Label Cover, serves as the starting point of almost all known NP-hardness of approximation (see e.g., [67–69]). The technique in [59] can also be used to show inapproximability for Label Cover with strong running time lower bound of the form \( f(k) \cdot N^{\Omega(k)} \) [70]. Due to known reductions, this has numerous consequences. For example, it implies, assuming Gap-ETH, that approximating \( k \)-EVEN SET to within any factor less than two cannot be done in \( f(k) \cdot N^{o(k)} \) time, considerably improving the lower bound mentioned in the previous subsection.

3.2.3. Inapproximability of \( k \)-Biclique and Densest \( k \)-Subgraph

While PIH (or equivalently the MULTICOLORED DENSEST \( k \)-SUBGRAPH problem) can serve as a starting point for hardness of approximation of many problems, there are some problems for which not even a constant factor hardness is known under PIH, but strong inapproximability results can be obtained via Gap-ETH. We will see two examples of this here.

First is the \( k \)-BICLIQUE problem. Recall that in this problem, we are given a bipartite graph and we would like to determine whether there is a complete bipartite subgraph of size \( k \). As stated earlier in the previous subsection, despite its close relationship to \( k \)-CLIQUE, \( k \)-BICLIQUE turned out to be a much more challenging problem to prove intractibility and even its W[1]-hardness was only shown recently [23]. This difficulty is corroborated by its approximability status in the classical (non-parameterized) regime; while CLIQUE is long known to be NP-hard to approximate to within \( N^{1-o(1)} \) factor [71], BICLIQUE is not even known to be NP-hard to approximate to within say 1.01 factor [72–75]. With this in mind, it is perhaps not a surprise that \( k \)-BICLIQUE is not known to be hard to approximate under PIH. Nonetheless, when we assume Gap-ETH, we can in fact prove a very strong hardness of approximation for the problem:

**Theorem 7 ([39]).** Assuming Gap-ETH, there is no \( o(k) \cdot \text{FPT-approximation for } k \)-BICLIQUE.

Note that, similar to \( k \)-CLIQUE, a \( k \)-approximation for BICLIQUE can be easily achieved by outputting a single edge. Hence, in terms of the inapproximability ratio, the above result is tight.

Due to its technicality, we only sketch an outline of the proof of Theorem 7 here. Firstly, the reduction starts by constructing a graph that is similar (but not the same) to that of \( k \)-Clique that we describe above. The main properties of this graph is that (i) in the YES case where the formula is satisfiable, the graph contains many copies of \( k \)-BICLIQUE, and (ii) in the NO case where the formula is not even \( (1-\delta) \)-satisfiable, the graph contains few copies of \( g \)-BICLIQUE. The construction and these properties were in fact shown in [76]. In [39], it was observed that, if we subsample the graph by keeping each vertex independently with probability \( p \) for an appropriate value of \( p \), then (i) ensures
that at least one of the \( k \)-BICLIQUE survives the subsampling, whereas (ii) ensures that no \( g \)-BICLIQUE survives. This indeed gives the claimed result in the above theorem.

We remark that, while Theorem 7 seems to resolve the approximability of \( k \)-BICLIQUE, there is still one aspect that is not yet completely understood: the running time lower bound. To demonstrate this, recall that, for \( k \)-CLIQUE, the reduction that gives hardness of \( k \)-VS-\( g \) CLIQUE has size \( 2^{O(n/g)} \); this means that we have a running time lower bound of \( f(k) \cdot N^{O(g)} \) on the problem. This is of course tight, because we can determine whether a graph has a \( g \)-clique in \( N^{O(g)} \) time. However, for \( k \)-BICLIQUE, the known reduction that gives hardness for \( k \)-VS-\( g \) BICLIQUE has size \( 2^{O(n/\sqrt{g})} \). This results in a running time lower bound of only \( f(k) \cdot N^{O(\sqrt{g})} \). Specifically, for the most basic setting of constant factor approximation, Theorem 7 only rules out algorithms with running time \( f(k) \cdot N^{o(\sqrt{g})} \). Hence, an immediate question here is:

**Open Question 8.** Is there an \( f(k) \cdot N^{o(k)} \)-time algorithm that approximates \( k \)-BICLIQUE to within a constant factor?

To put things into perspective, we note that, even for exact algorithms for \( k \)-BICLIQUE, the best running time lower bound is still \( f(k) \cdot N^{O(\sqrt{g})} \) \[23\] (under any reasonable complexity assumption). This means that, to answer Question 8, one has to first settle the best known running time lower bound for exact algorithms, which would already be a valuable contribution to the understanding of the problem.

Let us now point out an interesting consequence of Theorem 7 for the \( DENSEST k\)-SUBGRAPH problem. This is the "uncolored" version of the MULTICOLORED \( DENSEST k\)-SUBGRAPH problem as defined above, where there are no partitions \( V_1, \ldots, V_k \) and we can pick any \( k \) vertices in the input graph \( G \) with the objective of maximizing the number of induced edges. The approximability status of \( DENSEST k\)-SUBGRAPH very much mirrors that of \( k \)-BICLIQUE. Namely, in the parameterized setting, \( PIH \) is not known to imply hardness of approximation for \( DENSEST k\)-SUBGRAPH. Furthermore, in the classic (non-parameterized) setting, \( DENSEST k\)-SUBGRAPH is not known \[72,76–79\] to be NP-hard to approximate even to within a factor of say 1.01. Despite these, Gap-ETH does give a strong inapproximability for \( DENSEST k\)-SUBGRAPH, as stated below:

**Theorem 8 \([39]\).** Assuming Gap-ETH, there is no \( k^{o(1)} \)-FPT-approximation for \( DENSEST k\)-SUBGRAPH.

In fact, the above result is a simple consequence of Theorem 7. To see this, recall the following classic result in extremal graph theory commonly referred to as the Kövári–Sós–Turán (KST) Theorem \[80\]: any \( k \)-vertex graph that does not contain a \( g \)-biclique as a subgraph has density at most \( O(k^{-1/g}) \). Now, the hardness for \( k \)-BICLIQUE from Theorem 7 tells us that there is no FPT time algorithm that can distinguish between the graph containing \( k \)-biclique from one that does not contain \( g \)-biclique for any \( g = \omega(1) \). When the graph contains a \( k \)-biclique, we have a \( k \)-vertex subgraph with density (at least) 1/2. On the other hand, when the graph does not even contain a \( g \)-biclique, the KST Theorem ensures us that any \( k \)-vertex subgraph has density at most \( O(k^{1/g}) \). This indeed gives a gap of \( O(k^{1/g}) \) in terms of approximation \( DENSEST k\)-Subgraph and finishes the proof sketch for Theorem 8.

Unfortunately, Theorem 8 does not yet resolve the FPT approximability of \( DENSEST k\)-SUBGRAPH. In particular, while the hardness is only of the form \( k^{o(1)} \), the best known algorithm (which is the same as that of the multicolored version discussed above) only gives an approximation ratio of \( k \). Hence, we may ask whether this can be improved:

**Open Question 9.** Is there an \( o(k) \)-FPT-approximation algorithm for \( DENSEST k\)-SUBGRAPH?

This should be contrasted with Theorem 6, for which the FPT approximability of MULTICOLORED \( DENSEST k\)-SUBGRAPH is essentially resolved (up to lower order terms).
4. Algorithms

In this section we survey some of the developments on the algorithmic side in recent years. The organization of this section is according to problem types. We begin with basic packing and covering problems in Sections 4.1 and 4.2. We then move on to clustering in Section 4.3, network design in Section 4.4, and cut problems in Section 4.5. In Section 4.6 we present width reduction problems.

The algorithms in the above mentioned subsections compute approximate solutions to problems that are W[1]-hard. Therefore it is necessary to approximate, even when using parameterization. However, one may also aim to obtain faster parameterized runtimes than the known FPT algorithms, by sacrificing in the solution quality. We present some results of this type in Section 4.7.

4.1. Packing Problems

For a packing problem the task is to select as many combinatorial objects of some mathematical structure (such as a graph or a set system) as possible under some constraint, which restricts some objects to be picked if others are. A basic example is the INDEPENDENT SET problem, for which a maximum sized set of vertices of a graph needs to be found, such that none of them are adjacent to each other.

4.1.1. Independent Set

The INDEPENDENT SET problem is notoriously hard in general. Not only is there no polynomial time $n^{1-\epsilon}$-approximation algorithm [81] for any constant $\epsilon > 0$, unless P = NP, but also, under Gap-ETH, no $g(k)$-approximation can be computed in $f(k)n^{O(1)}$ time [39] for any computable functions $f$ and $g$, where $k$ is the solution size. On the other hand, for planar graphs a PTAS exists [82]. Hence a natural question is how the problem behaves for graphs that are “close” to being planar.

One way to generalize planar graphs is to consider minor-free graphs, because planar graphs are exactly those excluding $K_5$ and $K_{3,3}$ as minors. When parameterizing by the size of an excluded minor, the INDEPENDENT SET problem is paraNP-hard, since the problem is NP-hard on planar graphs [83]. Nevertheless a PAS can be obtained for this parameter [84].

**Theorem 9** ([84,85]). Let $H$ be a fixed graph. For $H$-minor-free graphs, INDEPENDENT SET admits an $(1 + \epsilon)$-approximation algorithm that runs in $f(H,\epsilon)n^{O(1)}$ time for some function $f$.

This result is part of the large framework of “bidimensionality theory” where any graph in an appropriate minor-closed class has treewidth bounded above in terms of the problem’s solution value, typically by the square root of that value. These properties lead to efficient, often subexponential, fixed-parameter algorithms, as well as polynomial-time approximation schemes, for bidimensional problems in many minor-closed graph classes. The bidimensionality theory is based on algorithmic and combinatorial extensions to parts of the Robertson–Seymour Graph Minor Theory, in particular initiating a parallel theory of graph contractions. The foundation of this work is the topological theory of drawings of graphs on surfaces. We refer the reader to the survey of [86] and more recent papers [85,87,88].

A different way to generalize planar graphs is to consider a planar deletion set, i.e., a set of vertices in the input graph whose removal leaves a planar graph. Taking the size of such a set as a parameter, INDEPENDENT SET is again paraNP-hard [83]. However, by first finding a minimum sized planar deletion set, then guessing the intersection of this set with the optimum solution to INDEPENDENT SET, and finally using the PTAS for planar graphs [82], a PAS can be obtained parameterized by the size of a planar deletion set [8].

**Theorem 10** ([8]). For the INDEPENDENT SET problem a $(1 + \epsilon)$-approximation can be computed in $2^k n^{O(1/\epsilon)}$ time for any $\epsilon > 0$, where $k$ is the size of a minimum planar deletion set.
Ideas using linear programming allow us to generalize and handle larger noise at the expense of worse dependence on $\epsilon$. Bansal et al. [89] showed that given a graph obtained by adding $\delta n$ edges to some planar graph, one can compute a $(1 + O(\epsilon + \delta))$-approximate independent set in time $n^{O(1/\epsilon^4)}$, which is faster than the $2^k n^{O(1/\epsilon)}$ running time of Theorem 10 for large $k = \delta n$. Magen and Moharrami [90] showed that for every graph $H$ and $\epsilon > 0$, given a graph $G = (V, E)$ that can be made $H$-minor-free after at most $\delta n$ deletions and additions of vertices or edges, the size of the maximum independent set can be approximately computed within a factor $(1 + \epsilon + O(\delta |H| \sqrt{\log |H|}))$ in time $n^{f(\epsilon, |H|)}$. Note that this algorithm does not find an independent set. Recently, Demaine et al. [91] presented a general framework to obtain better approximation algorithms for various problems including INDEPENDENT SET and CHROMATIC NUMBER, when the input graph is close to well-structured graphs (e.g., bounded degeneracy, degree, or treewidth).

It is also worth noting here that INDEPENDENT SET problem can be generalized to the $d$-SCATTERED SET problem where we are given an (edge-weighted) graph and are asked to select at least $k$ vertices, so that the distance between any pair is at least $d$ [92]. Recently in [93] some lower and upper bounds on the approximation of the $d$-SCATTERED SET problem have been provided.

A special case of INDEPENDENT SET is the INDEPENDENT SET OF RECTANGLES problem, where a set of axis-parallel rectangles is given in the two-dimensional plane, and the task is to find a maximum sized subset of non-intersecting rectangles. This is a special case, since pairwise intersections of rectangles can be encoded by edges in a graph for which the vertices are the rectangles. Parameterized by the solution size, the problem is W[1]-hard [94], and while a QPTAS is known [95], it is a challenging open question whether a PTAS exists. It was shown [96] however that both a PAS and a PSAKS exist for INDEPENDENT SET OF RECTANGLES parameterized by the solution size, even for the weighted version.

The runtime of this PAS is $f(k, \epsilon)n^{g(\epsilon)}$ for some functions $f$ and $g$, where $k$ is the solution size. Note that the dependence on $\epsilon$ in the degree of the polynomial factor of this algorithm cannot be removed, unless FPT=W[1], since any efficient PAS with runtime $f(k, \epsilon)n^{O(1)}$ could be used to compute the optimum solution in FPT time by setting $\epsilon$ to $\frac{1}{\epsilon + 1}$ in the W[1]-hard unweighted version of the problem [94]. However, in the so-called shrinking model an efficient PAS can be obtained [97] for INDEPENDENT SET OF RECTANGLES. The parameter in this case is a factor $0 < \delta < 1$ by which each rectangle is shrunken before computing an approximate solution, which is compared to the optimum solution without shrinking.

**Theorem 11 ([96,97]).** For the INDEPENDENT SET OF RECTANGLES problem a $(1 + \epsilon)$-approximation can be computed in $k^{O(k/\delta^2)}n^{O(1/\epsilon^6)}$ time for any $\epsilon > 0$, where $k$ is the size of the optimum solution, or in $f(\delta, \epsilon)n^{O(1)}$ time for some computable function $f$ and any $\epsilon > 0$ and $0 < \delta < 1$, where $\delta$ is the shrinking factor. Moreover, a $(1 + \epsilon)$-approximate kernel with $k^{O(1/\epsilon^6)}$ rectangles can be computed in polynomial time.

Another special case of INDEPENDENT SET is the INDEPENDENT SET ON UNIT DISK GRAPH problem, where given set of $n$ unit disks in the Euclidean plane, the task is to determine if there exists a set of $k$ non-intersecting disks. The problem is NP-hard [98] but admits a PTAS [99]. Marx [94] showed that, when parameterized by the solution size, the problem is W[1]-hard; this also rules out EPTAS (and even efficient PAS) for the problem, assuming FPT $\neq$ W[1]. On the other hand, in [100] the authors give an FPT algorithm for a special case of INDEPENDENT SET ON UNIT DISK GRAPH when there is a lower bound on the distance between any pair of centers.

4.1.2. Vertex Coloring

A problem related to INDEPENDENT SET is the VERTEX COLORING problem, for which the vertices need to be colored with integer values, such that no two adjacent vertices have the same color (which means that each color class forms an independent set in the graph). The task is to minimize the number of used colors. For planar graphs the problem has a polynomial time 4/3-approximation algorithm [8] via the celebrated Four Color Theorem, and a better approximation is not possible in
polynomial time [101]. Using this algorithm, a 7/3-approximation can be computed in FPT time when parameterizing by the size of a planar deletion set [8]. When generalizing planar graphs by excluding any fixed minor, and taking its size as the parameter, a 2-approximation can be computed in FPT time [102]. Due to the NP-hardness for planar graphs [101], neither of these two parameterizations admits a PAS, unless P = NP.

**Theorem 12 ([8,102]).** For the VERTEX COLORING problem

- a 7/3-approximation can be computed in \(k^k n^{O(1)}\) time, where \(k\) is the size of a minimum planar deletion set, and
- a 2-approximation can be computed in \(f(k)n^{O(1)}\) time for some function \(f\), where \(k\) is the size of an excluded minor of the input graph.

One way to generalize VERTEX COLORING is to see each color class as an induced graph of degree 0. The DEFECTIVE COLORING problem [103] correspondingly asks for a coloring of the vertices, such that each color class induces a graph of maximum degree \(\Delta\), for some given \(\Delta\). The aim again is to minimize the number of used colors. In contrast to VERTEX COLORING, the DEFECTIVE COLORING problem is W[1]-hard [104] parameterized by the treewidth. This parameter measures how “tree-like” a graph is, and is defined as follows.

**Definition 1.** A tree decomposition of a graph \(G = (V, E)\) is a tree \(T\) for which every node is associated with a bag \(X \subseteq V\), such that the following properties hold:

1. the union of all bags is the vertex set \(V\) of \(G\),
2. for every edge \((u, v)\) of \(G\), there is a node of \(T\) for which the associated bag contains \(u\) and \(v\), and
3. for every vertex \(u\) of \(G\), all nodes of \(T\) for which the associated bags contain \(u\), induce a connected subtree of \(T\).

The width of a tree decomposition is the size of the largest bag minus 1 (which implies that a tree has a decomposition of width 1 where each bag contains the endpoints of one edge). The treewidth of a graph is the smallest width of any of its tree decompositions.

Treewidth is fundamental parameter of a graph and will be discussed more elaborately in Section 4.6.1. However, it is worth mentioning here that VERTEX COLORING is in FPT when parameterized by treewidth.

The strong polynomial-time approximation lower bound of \(n^{1-\varepsilon}\) for VERTEX COLORING [81] naturally carries over to the more general DEFECTIVE COLORING problem. A much improved approximation factor of 2 is possible though in FPT time if the parameter is the treewidth [104]. It can be shown however, that a PAS is not possible in this case, as there is no \((3/2 - \varepsilon)\)-approximation algorithm for any \(\varepsilon > 0\) parameterized by the treewidth [104], unless FPT = W[1]. A natural question is whether the bound \(\Delta\) of DEFECTIVE COLORING can be approximated instead of the number of colors. For this setting, a bicriteria PAS parameterized by the treewidth exists [104], which computes a solution with the optimum number of colors where each color class induces a graph of maximum degree at most \((1 + \varepsilon)\Delta\).

**Theorem 13 ([104]).** For the DEFECTIVE COLORING problem, given a tree decomposition of width \(k\) of the input graph,

- a solution with the optimum number of colors where each color class induces a graph of maximum degree \((1 + \varepsilon)\Delta\) can be computed in \((k/\varepsilon)^{O(k)} n^{O(1)}\) time for any \(\varepsilon > 0\),
- a 2-approximation (of the optimum number of colors) can be computed in \(k^{O(k)} n^{O(1)}\) time, but
- no \((3/2 - \varepsilon)\)-approximation (of the optimum number of colors) can be computed in \(f(k)n^{O(1)}\) time for any \(\varepsilon > 0\) and computable function \(f\), unless FPT = W[1].
The algorithms of the previous theorem build on the techniques of [105] using approximate addition trees in combination with dynamic programs that yield XP algorithms for these problems. This technique can be applied to various problems (cf. Section 4.2), including a different generalization of VERTEX COLORING called EQUITABLE COLORING. Here the aim is to color the vertices of a graph with as few colors as possible, such that every two adjacent vertices receive different colors, and all color classes contain the same number of vertices. It is a generalization of VERTEX COLORING, since one may add a sufficiently large independent set (i.e., a set of isolated vertices) to a graph such that the number of colors needed for an optimum VERTEX COLORING solution is the same as for an optimum EQUITABLE COLORING solution.

The EQUITABLE COLORING problem is W[1]-hard even when combining the number of colors needed and the treewidth of the graph as parameters [106]. On the other hand, a PAS exists [105] if the parameter is the cliquewidth of the input graph. This is a weaker parameter than treewidth, as the cliquewidth of a graph is bounded as a function of its treewidth. However, while bounded treewidth graphs are sparse, cliquewidth also allows for dense graphs (such as complete graphs). Formally, a graph of cliquewidth $\ell$ can be constructed using the following recursive operations using $\ell$ labels on the vertices:

1. Introduce($x$): create a graph containing a singleton vertex labelled $x \in \{1, \ldots, \ell\}$.
2. Union($G_1, G_2$): return the disjoint union of two vertex-labelled graphs $G_1$ and $G_2$.
3. Join($G, x, y$): add all edges connecting a vertex of label $x$ with a vertex of label $y$ to the vertex-labelled graph $G$.
4. Rename($G, x, y$): change the label of every vertex of $G$ with label $x$ to $y \in \{1, \ldots, \ell\}$.

A cliquewidth expression with $\ell$ labels is a recursion tree describing how to construct a graph using the above four operations using only labels from the set $\{1, \ldots, \ell\}$. Notice that the cliquewidth of a complete graph is two and therefore we have graphs of bounded clique-width but unbounded treewidth. As stated earlier the cliquewidth of a graph is bounded above exponentially in its treewidth and this dependence is tight for some graph families [107].

The PAS for EQUITABLE COLORING will compute a coloring using at most $k$ colors such that the ratio between the sizes of any two color classes is at most $1 + \epsilon$. In this sense it is a bicriteria approximation algorithm.

**Theorem 14 ([105]).** For the EQUITABLE COLORING problem, given a cliquewidth expression with $\ell$ labels for the input graph, a solution with optimum number of colors where the ratio between the sizes of any two color classes is at most $1 + \epsilon$, can be computed in $(k/\epsilon)^{O(k)}n^{O(1)}$ time [108,109] for any $\epsilon > 0$, where $k$ is the optimum number of colors.

A variant of VERTEX COLORING is the MIN SUM COLORING problem, where, instead of minimizing the number of colors, the aim is to minimize the sum of (integer) colors, where the sum is taken over all vertices. This problem is FPT parameterized by the treewidth [110], but the related MIN SUM EDGE COLORING problem is NP-hard [111] on graphs of treewidth 2 (while being polynomial time solvable on trees [112]). For this problem the edges need to be colored with integer values, so that no two edges sharing a vertex have the same color, and the aim again is to minimize the total sum of colors. Despite being APX-hard [111] and also paraNP-hard for parameter treewidth, MIN SUM EDGE COLORING admits a PAS for this parameter [113].

**Theorem 15 ([113]).** For the MIN SUM EDGE COLORING problem a $(1 + \epsilon)$-approximation can be computed in $f(k, \epsilon)n$ time for any $\epsilon > 0$, where $k$ is the treewidth of the input graph.

### 4.1.3. Subgraph Packing

A special family of packing problems can be obtained by subgraph packing. Let $H$ be a fixed “pattern” graph. The $H$-PACKING problem, given the “host” graph $G$, asks to find the maximum
number of vertex-disjoint copies of \( H \). One can also let \( H \) be a family of graphs and ask the analogous problem. There is another choice whether each copy of \( H \) is required to be an induced subgraph or a regular subgraph. We focus on the regular subgraph case here.

When \( H \) is a single graph with \( k \) vertices, a simple greedy algorithm that finds an arbitrary copy of \( H \) and adds it to the packing, guarantees a \( k \)-approximation in time \( f(H, n) \cdot n \). Here \( f(H, n) \) denotes the time to find a copy of \( H \) in an \( n \)-vertex graph. Following a general result for \( k \)-SET PACKING, a \((k + 1 + \epsilon)/3\)-approximation algorithm that runs in polynomial time for fixed \( k, \epsilon \) exists [114]. When \( H \) is 2-vertex-connected or a star graph, even for fixed \( k \), it is \( \text{NP} \)-hard to approximate the problem better than a factor \( \Omega(k/\text{polylog}(k)) \) [115]. There is no known connected \( H \) that admits an \( \text{FPT} \) (or even \( \text{XP} \)) algorithm achieving a \( k^{1-\delta} \)-approximation for some \( \delta > 0 \); in particular, the parameterized approximability of \( k \)-PATH PACKING is wide open. It is conceivable that \( k \)-PATH PACKING admits a parameterized \( \omega(k) \)-approximation algorithm, given an \( O(\log k) \)-approximation algorithm for \( k \)-PATH DELETION [116] and an improved kernel for INDUCED \( P_3 \) PACKING [117].

When \( H \) is the family of all cycles, the problem becomes the VERTEX CYCLE PACKING problem, for which the largest number of vertex-disjoint cycles of a graph needs to be found. No polynomial time \( O(\log^{1/2-\epsilon} n) \)-approximation is possible for this problem [118] for any \( \epsilon > 0 \), unless every problem in \( \text{NP} \) can be solved in randomized quasi-polynomial time. Furthermore, despite being \( \text{FPT} \) [119] parameterized by the solution size, VERTEX CYCLE PACKING does not admit any polynomial-sized exact kernel for this parameter [120], unless \( \text{NP} \subseteq \text{coNP} / \text{poly} \). Nevertheless, a PSAKS can be found [18].

**Theorem 16** ([18]). *For the VERTEX CYCLE PACKING problem, a \((1 + \epsilon)\)-approximate kernel of size \( k^{O(1/\epsilon \log k)} \) can be computed in polynomial time, where \( k \) is the solution size.*

### 4.1.4. Scheduling

Yet another packing problem on graphs, which, however, has applications in scheduling and bandwidth allocation, is the UNSPLITTABLE FLOW ON A PATH problem. Here a path with edge capacities is given together with a set of tasks, each of which specifies a start and end vertex on the path and a demand value. The goal is to find the largest number of tasks such that for each edge on the path the total demand of selected tasks for which the edge lies between its start and end vertex, does not exceed the capacity of the edge. This problem admits a QPTAS [121], but it remains a challenging open question whether a PTAS exists. When parameterizing by the solution size, UNSPLITTABLE FLOW ON A PATH is \( \text{W}[1] \)-hard [122]. However a PAS exists [122] for this parameter.

**Theorem 17** ([122]). *For the UNSPLITTABLE FLOW ON A PATH problem a \((1 + \epsilon)\)-approximation can be computed in \( 2^{O(k \log k)} n^g(\epsilon) \) time for some computable function \( g \) and any \( \epsilon > 0 \), where \( k \) is the solution size.*

Another scheduling problem is FLOW TIME SCHEDULING, for which a set of jobs is given, each of which is specified by a processing time, a release date, and a weight. The jobs need to be scheduled on a given number of machines, such that no job is processed before its release date and a job only runs on one machine at a time. Given a schedule, the flow time of a job is the weighted difference between its completion time and release date, and the task for the FLOW TIME SCHEDULING problem is to minimize the sum of all flow times. Two types of schedules are distinguished: in a preemptive schedule a job may be interrupted on one machine and then resumed on another, while in a non-preemptive schedule every job runs on one machine until its completion once it was started. If pre-emptive schedules are allowed, FLOW TIME SCHEDULING has no polynomial time \( O(\log^{1-\epsilon} p) \)-approximation algorithm [123], unless \( P = \text{NP} \), where \( p \) is the maximum processing time. For the more restrictive non-preemptive setting, no \( O(n^{1/2-\epsilon}) \)-approximation can be computed in polynomial time [124], unless \( P = \text{NP} \), where \( n \) is the number of jobs. The latter lower bound is in fact even valid for only one machine, and thus parameterizing FLOW TIME SCHEDULING by the number of machines will not yield any better approximation ratio in this setting. A natural parameter for FLOW TIME SCHEDULING is the maximum
over all processing times and weights of the given jobs. It is not known whether the problem is FPT or \( W[1] \)-hard for this parameter. However, when combining this parameter with the number of machines, a PAS can be obtained \([125]\) despite the strong polynomial time approximation lower bounds.

**Theorem 18** \([125]\). For the Flow Time Scheduling problem a \((1 + \varepsilon)\)-approximation can be computed in \((mk)^{O(mk^2/\varepsilon)}n^{O(1)}\) time in the preemptive setting, and in \((mk/\varepsilon)^{O(mk^2)}n^{O(1)}\) time in the non-preemptive setting, for any \( \varepsilon > 0 \), where \( m \) is the number of machines and \( k \) is an upper bound on every processing time and weight.

4.2. Covering Problems

For a covering problem the task is to select a set of \( k \) combinatorial objects in a mathematical structure, such as a graph or set system (i.e., hypergraph), under some constraints that demands certain other objects to be intersected/covered. A basic example is the Set Cover problem where we are given a set system, which is simply a collection of subsets of a universe. The goal is to determine whether there are \( k \) subsets whose union cover the whole universe.

There are two ways define optimization based on covering problems. First, we may view the covering demands as strict constraints and aim to find a solution that minimize the constraint/cost while covering all objects (i.e., relaxing the size-\( k \) constraint); this results in a minimization problem. Second, we may view the size constraint as a strict constraint and aim to find a solution that covers as many objects as possible; this results in a maximization problem. We divide our discussion mainly into two parts, based on these two types of optimization problems. In Section 4.2.3, we discuss problems related to covering that fall into neither category.

4.2.1. Minimization Variants

We start out discussion with the minimization variants. For brevity, we overload the problem name and use the same name for the minimization variant (e.g., we use Set Cover instead of the more cumbersome Min Set Cover). Later on, we will use different names for the maximization versions; hence, there will be no confusion.

**Set Cover, Dominating Set and Vertex Cover.** As discussed in detail in Section 3.1.2, Set Cover and equivalently Dominating Set are very hard to approximate in the general case. Hence, special cases where some constraints are placed on the set system are often considered. Arguably the most well-studied special case of Set Cover is the Vertex Cover problem, in which the set system is a graph. That is, we would like to find the smallest set of vertices such that every edge has at least one endpoint in the selected set (i.e., the edge is “covered”). Vertex Cover is well known to be FPT \([126]\) and admit a linear-size kernel \([127]\). A generalization of Vertex Cover on \( d \)-uniform hypergraph, where the input is now a hypergraph and the goal is to find the smallest set of vertices such that every hyperedge contain at least one vertex from the set, is also often referred to as \( d \)-Hitting Set in the parameterized complexity community. However, we will mostly use the nomenclature Vertex Cover on \( d \)-uniform hypergraph because many algorithms generalizes well from Vertex Cover in graphs to hypergraphs. Indeed, branching algorithms for Vertex Cover on graphs can be easily generalized to hypergraphs, and hence the latter is also FPT. Polynomial-size kernels are also known for Vertex Cover on \( d \)-uniform hypergraphs \([128]\).

While Vertex Cover both on graphs and \( d \)-uniform hypergraphs are already tractable, approximation can still help make algorithms even faster and kernels even smaller. We defer this discussion to Section 4.7.

**Connected Vertex Cover.** A popular variant of Vertex Cover that is the Connected Vertex Cover problem, for which the computed solution is required to induce a connected subgraph of the input. Just as Vertex Cover, the problem is FPT \([129]\). However, unlike Vertex Cover, Connected Vertex Cover does not admit a polynomial-time kernel \([130]\), unless \( \text{NP} \subseteq \text{coNP}/\text{poly} \). In spite of this, a PAS can be obtained \([131]\) even though the problem is \( W[2] \)-hard for this parameter. However, when combining this parameter with the number of machines, where the input is now a hypergraph and the goal is to find the smallest set of vertices such that every edge has at least one endpoint in the selected set (i.e., the edge is “covered”). Vertex Cover is well known to be FPT \([126]\) and admit a linear-size kernel \([127]\). A generalization of Vertex Cover on \( d \)-uniform hypergraph, where the input is now a hypergraph and the goal is to find the smallest set of vertices such that every hyperedge contain at least one vertex from the set, is also often referred to as \( d \)-Hitting Set in the parameterized complexity community. However, we will mostly use the nomenclature Vertex Cover on \( d \)-uniform hypergraph because many algorithms generalizes well from Vertex Cover in graphs to hypergraphs. Indeed, branching algorithms for Vertex Cover on graphs can be easily generalized to hypergraphs, and hence the latter is also FPT. Polynomial-size kernels are also known for Vertex Cover on \( d \)-uniform hypergraphs \([128]\).

While Vertex Cover both on graphs and \( d \)-uniform hypergraphs are already tractable, approximation can still help make algorithms even faster and kernels even smaller. We defer this discussion to Section 4.7.
**Theorem 19** ([18]). For any $\varepsilon > 0$, an $(1 + \varepsilon)$-approximate kernel with $k^{O(1/\varepsilon)}$ vertices can be computed in polynomial time.

The ideas behind [18] is quite neat and we sketch it here. There are two reduction rules: (i) if there exists a vertex with degree more than $\Delta := 1/\varepsilon$ just “select” the vertex and (ii) if we see a vertex with more than $k$ false twins, i.e., vertices with the same set of neighbors, then we simply remove it from the graph. An important observation for (i) is that, since we have to either pick the vertex or all $\geq \Delta$ neighbors anyway, we might as well just select it even in the second case because it affects the size of the solution by a factor of at most $1 + \frac{1}{\Delta} = 1 + \varepsilon$. For (ii), it is not hard to see that we either select one of the false twins or all of them; hence, if a vertex has more than $k$ false twins, then it surely cannot be in the optimal solution. Roughly speaking, these two observations show that this is an $(1 + \varepsilon)$-kernel. Of course, in the actual proof, “selecting” a vertex needs to be defined more carefully, but we will not do it here. Nonetheless, imagine the end step when we cannot apply these two reduction rules anymore. Essentially speaking, we end up with a graph where some (less than $(1 + \varepsilon)k$) vertices are marked as “selected” and the remaining vertices have degree at most $\Delta$. Now, every vertex is either inside the solution, or all of its neighbors must lie in the solution. There are only (at most) $k$ vertices in the first case. For the second case, note that these vertices have degree at most $\Delta$ and they have at most $k$ false twins, meaning that there are at most $k^{1+\Delta} = k^{1+1/\varepsilon}$ such vertices. In other words, the kernel is of size $k^{O(1/\varepsilon)}$ as desired. This constitutes the main ideas in the proof; let us stress again that the actual proof is of course more complicated than this since we did not define rule (i) formally.

Recently, Krithika et al. [131] considered the following structural parameters beyond the solution size: split deletion set, clique cover and cluster deletion set. In each case, the authors provide a PSAKS for the problem. We will not fully define these parameters here, but we note that the first parameter (split deletion set) is always no larger than the size of the minimum vertex cover of the graph. In another very recent work, Majumdar et al. [132] give a PSAKS for each of the following parameters, each of which is always no larger than the solution size: the deletion distance of the input graph to the class of cographs, the class of bounded treewidth graphs, and the class of all chordal graphs. Hence, these results may be viewed as a generalization of the aforementioned PSAKS from [18].

**Connected Dominating Set.** Similar to **Connected Vertex Cover**, the **Connected Dominating Set** problem is the variant of **Dominating Set** for which the solution additionally needs to induce a connected subgraph of the input graph. When placing no restriction on the input graph, the problem is as hard to approximate as **Dominating Set**. However, for some special classes of graphs, PSAKS or bi-PSAKS [133] are known; these include graphs with bounded expansion, nowhere dense graphs, and $d$-biclique-free graphs [134].

**Covering Problems parameterized by Graph Width Parameters.** Several works in literature also study the approximability of variants of **Vertex Cover** and **Dominating Set** parameterized by graph widths [105,135]. These variants include:

- **Power Vertex Cover** (PVC). Here, along with the input graph, each edge has an integer demand and we have to assign (power) values to vertices, such that each edge has at least one endpoint with a value at least its demand. The goal is to minimize the total assigned power. Note that this is generalizes of **Vertex Cover**, where edges have unit demands.

- **Capacitated Vertex Cover** (CVC). The problem is similar to **Vertex Cover**, except that each vertex has a capacity which limits the number of edges that it can cover. Once again, **Vertex Cover** is a special case of CVC where each vertex’s capacity is $\infty$.

- **Capacitated Dominating Set** (CDS). Analogous to CVC, this is a generalization of **Dominating Set** where each vertex has a capacity and it can only cover/dominat at most that many other vertices.
All problems above are FPT under standard parameter (i.e., the optimum) \[135,136\]. However, when parameterizing by the treewidth \[137\], all three problems become W[1]-hard \[135\]. (This is in contrast to Vertex Cover and Dominating Set, both of which admit straightforward dynamic programming FPT algorithms parameterized by treewidth.) Despite this, good FPT approximation algorithms are known for the problem. In particular, a PAS is known for PVC \[135\]. For CVC and CDS, a bicriteria PAS exists for the problem \[105\], which in this case computes a solution of size at most the optimum, so that no vertex capacity is violated by more than a factor of \(1 + \varepsilon\).

The approximation algorithms for CVC and CDS are results of a more general approach of Lampis \[105\]. The idea is to execute an “approximate” version of dynamic programming in tree decomposition instead of the exact version; this helps reduce the running time from \(n^{O(k)}\) to \((\log n/\varepsilon)^{O(\varepsilon)}\), which is FPT. The approach is quite flexible: several approximation for graphs problems including covering problems can be achieved via this method and it also applies to clique-width. Please refer to \[105\] for more details.

### Packing-Covering Duality and Erdős-Pósa Property

Given a set system \((V,C)\) where \(V\) is the universe and \(C = \{C_1, \ldots, C_m\}\) is a collection of subsets of \(V\), Hitting Set is the problem of computing the smallest \(S \subseteq V\) that intersects every \(C_i\), and Set Packing is the problem of computing the largest subcollection \(C' \subseteq C\) such that no two sets in \(C'\) intersect. It can also be observed that the optimal value for Hitting Set is at least the optimal value for Set Packing, while the standard LP relaxations for them (covering LP and packing LP) have the same optimal value by strong duality. Studying the other direction of the inequality (often called the packing-covering duality) for natural families of set systems has been a central theme in combinatorial optimization. The gap between the covering optimum and packing optimum is large in general (e.g., Dominating Set/Independent Set), but can be small for some families of set systems (e.g., s-t Cut/s-t Disjoint Paths and Vertex Cover/Matching especially in bipartite graphs).

One notion that has been important for both parameterized and approximation algorithms is the Erdős-Pósa property \[138\]. A family of set systems is said to have the Erdős-Pósa property when there is a function \(f : \mathbb{N} \rightarrow \mathbb{N}\) such that for any set system in the family, if the packing optimum is \(k\), the covering optimum is at most \(f(k)\). This immediately implies that the multiplicative gap between these two optima is at most \(f(k)/k\), and constructive proofs for the property for various set systems have led to \(f(k)/k\)-approximation algorithms. Furthermore, for some problems including Cycle Packing, the Erdős-Pósa property gives an immediate parameterized algorithm. We refer the reader to a recent survey \[139\] and papers \[119,140,141\].

The original paper of Erdős and Pósa \[138\] proved the property for set systems \((V,C)\) when there is an underlying graph \(G = (V,E)\) and \(C\) is the set of cycles, which corresponds to the pair Cycle Packing/Feedback Vertex Set; every graph either has at least \(k\) vertex-disjoint cycles or there is a feedback vertex set of size at most \(O(k \log k)\). Many subsequent papers also studied natural set systems arising from graphs where \(V\) is the set of vertices or edges and \(C\) denotes a collection of subgraphs of interest. For those set systems, Erdős-Pósa Properties are closely related to Set Packing introduced in Section 4.1.3 and \(F\)-Deletion problems introduced in Section 4.6.

#### 4.2.2. Maximization Variants

We now move on to the maximization variants of covering problems. To our knowledge, these covering problems are much less studied in the context of parameterized approximability compared to their minimization counterparts. In particular, we are only aware of works on the maximization variants of Set Cover and Vertex Cover, which are typically called Max \(k\)-Coverage and Max \(k\)-Vertex Cover respectively.

**Max \(k\)-Coverage.** Recall that here we are given a set system and the goal is to select \(k\) subsets whose union is maximized. It is well known that the simple greedy algorithm yields an \((1-\frac{1}{e})\)-approximation \[142\]. Furthermore, Fiege shows, in his seminal work \[29\], show that this is
tight: \(\left(\frac{1}{\epsilon} - \epsilon\right)\)-approximation is NP-hard for any constant \(\epsilon > 0\). In fact, recently it has been shown that this inapproximability applies also to the parameterized setting. Specifically, under Gap-ETH, \(\left(\frac{1}{\epsilon} - \epsilon\right)\)-approximation cannot be achieved in FPT time [143] or even \(f(k) \cdot n^{o(k)}\) time [70]. In other words, the trivial algorithm is tight in terms of running time, the greedy algorithm is tight in terms of approximation ratio, and there is essentially no trade-off possible between these two extremes. We remark here that this hardness of approximation is also the basis of hardness for \(k\)-MEANS [143] (see Section 4.3).

Due to the strong inapproximability result for the general case of \(\text{MAX} \ k\)-\(\text{COVERAGE}\), different parameters have to be considered in order to obtain a PAS for \(\text{MAX} \ k\)-\(\text{COVERAGE}\). An interesting positive result here is when the parameters are \(k\) and the VC dimension of the set system, for which a PAS exists while the exact version of the problem is W[1]-hard [144].

**Max \(k\)-\text{Vertex Cover}**. Another special case of \(\text{MAX} \ k\)-\(\text{COVERAGE}\) is the restriction when each element belongs to at most \(d\) subsets in the system. This corresponds exactly to the maximization variant of the \(\text{VERTEX COVER}\) problem on \(d\)-uniform hypergraph, which will refer to as \(\text{MAX} \ k\)-\(\text{VERTEX COVER}\). Note here that, for such set systems, their VC-dimensions are also bounded by \(\log d + 1\) and hence the aforementioned PAS of [144] applies here as well. Nonetheless, \(\text{MAX} \ k\)-\(\text{VERTEX COVER}\) admits a much simpler PAS (and even PSAKS) compared to \(\text{MAX} \ k\)-\(\text{COVERAGE}\) parameterized by \(k\) and VC-dimension, as we will discuss more below.

\(\text{MAX} \ k\)-\(\text{VERTEX COVER}\) was first studied in the context of parameterized complexity by Guo et al. [145] who showed that the problem is W[1]-hard. Marx, in his survey on parameterized approximation algorithms [8], gave a PAS for the problem with running time \(2^{O(k^3/\epsilon)}\). Later, Lokshtanov et al. [18] shows that Marx’s approach can be used to give a PSAKS of size \(O(k^5/\epsilon^2)\). Both of these results mainly focus on graphs. Later, Skowron and Faliszewski [146–148] gave a more general argument that both works generally for any \(d\)-uniform hypergraph and improves the running time and kernel size:

**Theorem 20** ([146]). For the \(\text{MAX} \ k\)-\(\text{VERTEX COVER}\) problem in \(d\)-uniform hypergraphs, a \((1 + \epsilon)\)-approximation can be computed in \(O^* \left( (d/\epsilon)^k \right) \) time for any \(\epsilon > 0\). Moreover, an \((1 + \epsilon)\)-approximate kernel with \(O(dk/\epsilon)\) vertices can be computed in polynomial time.

The main idea of the above proof is simple and elegant, and hence we will include it here. For convenience, we will only discuss the graph case, i.e., \(d = 2\). It suffices to just give the \(O(k/\epsilon)\)-vertex kernel; the PAS immediately follows by running the brute force algorithm on the output instance from the kernel. The kernel is as simple as it gets: just keep \(2k/\epsilon\) vertices with highest degrees and throw the remaining vertices away! Note that there is a subtle point here, which is that we do not want to throw away the edges linking from the kept vertices to the remaining vertices. If self-loops are allowed in a graph, this is not an issue since we may just add a self-loop to each vertex for each edge adjacent to it with the other endpoint being discarded. When self-loops are not allowed, it is still possible to overcome this issue but with slightly larger kernel; we refer the readers to Section 3.2 of [147] for more detail.

Having defined the kernel, let us briefly discuss the intuition as to why it works. Let \(V_{2k/\epsilon}\) denote the set of \(2k/\epsilon\) highest-degree vertices. The main argument of the proof is that, if there is an optimal solution \(S\), then we may modify it to be entirely contained in \(V_{2k/\epsilon}\) while preserving the number of covered edges to within \((1 + \epsilon)\) factor. The modification is simple: for every vertex that is outside of \(V_{2k/\epsilon}\) we replace it with a random vertex from \(V_{2k/\epsilon}\). Notice here that we always replace a vertex with a higher-degree vertex. Naturally, this should be good in terms of covering more edges, but there is a subtle point here: it is possible that the high degree vertices are “double counted” if a particular edge is covered by both endpoints. The size \(2k/\epsilon\) is selected exactly to combat this issue; since the set is large enough, “double counting” is rare for random vertices. This finishes our outline for the intuition.
We end by remarking that \( \text{MAX } k\text{-VERTEX COVER} \) on graphs is already APX-hard \cite{149}, and hence the PASes mentioned above once again demonstrate additional power of FPT approximation algorithms over polynomial-time approximation algorithms.

4.2.3. Other Related Problems

There are several other covering-related problems that do not fall into the two categories we discussed so far. We discuss a couple such problems below.

\textbf{Min } k\text{-Uncovered.} The first is the \textsc{Min } \( k\text{-UNCOVERED} \) problem, where the input is a set system and we would like to select \( k \) sets as to minimize the number of uncovered elements. When we are concerned with exact solutions, this is of course the \textsc{Set Cover}. However, the optimization version becomes quite different from \textsc{Max } \( k\text{-Coverage} \). In particular, since it is hard to determine whether we can find \( k \) subsets that cover the whole universe, the problem is not approximable at all in the general case. However, if restrict ourselves to graphs and hypergraphs (for which we refer to the problem as \textsc{Min } \( k\text{-Vertex Cover} \)), it is possible to get a (randomized) PAS for the problem \cite{146}:

\begin{theorem}[(\cite{146})]
For the \textsc{Min } \( k\text{-Vertex Uncovered} \) problem in \( d\text{-uniform hypergraphs} \), a \((1 + \varepsilon)\)-approximation can be computed in \( O^\ast \left( \left( d/\varepsilon \right)^{k} \right) \) time for any \( \varepsilon > 0 \).
\end{theorem}

The algorithm is based on the following simple randomized branching: pick a random uncovered element and branch on all possibilities of selecting a subset that contains it. Notice that since an element belongs to only \( d \) subsets, the branching factor is at most \( d \). The key intuition in the approximation proof is that, when the number of elements we have covered so far is still much less than that in the optimal solution, there is a relatively large probability (i.e., \( \varepsilon \)) that the random element is covered in the optimal solution. If we always pick such a “good” element in most branching steps, then we would end up with a solution close to the optimum. Skowron and Faliszewski \cite{146} formalizes this intuition by showing that the algorithm outputs an \((1 + \varepsilon)\)-approximate solution with probability roughly \( \varepsilon^k \).

Hence, by repeating the algorithm \( (1/\varepsilon)^k \) time, one arrives at the claimed PAS. To the best of our knowledge, it is unknown whether a PSAKS exists for the problem.

\textbf{Min } k\text{-Coverage.} Another variant of the \textsc{Set Cover} problem studied is \textsc{Min } \( k\text{-Coverage} \) \cite{150–152}, where we would like to select \( k \) subsets that minimizes the number of covered elements. We stress here that this problem is not a relaxation of \textsc{Set Cover} but rather is much more closely related to graph expansion problems (see \cite{151}).

It is known that, when there is no restriction on the input set system, the problem is (up to a polynomial factor) as hard to approximate as the \textsc{Denest } \( k\text{-Subgraph} \) problem \cite{150}. Hence, by the inapproximability of the latter discussed earlier in the survey (Theorem 8), we also have that there is no \( k^{o(1)} \)-approximation algorithm for the problem that runs in FPT time.

Once again, the special case that has been studied in literature is when the input set system is a graph, in which case we refer to the problem as \textsc{Min } \( k\text{-Vertex Cover} \). Gupta, Lee, and Li \cite{153,154} used the technique of Marx \cite{8} to give a PAS for the problem with running time \( O^\ast \left( (k/\varepsilon)^{O(k)} \right) \). The running time was later improved in \cite{147} to \( O^\ast \left( (1/\varepsilon)^{O(k)} \right) \). The algorithm there is again based on branching, but the rules are more delicate and we will not discuss them here. An interesting aspect to note here is that, while both \textsc{Max } \( k\text{-Vertex Cover} \) and \textsc{Min } \( k\text{-Vertex Cover} \) have PSAKS of the (asymptotically) same running time, the former admits a PSAKS whereas the latter does not (assuming a variant of the Small Set Expansion Conjecture) \cite{147}.

To the best of our knowledge, \textsc{Min } \( k\text{-Vertex Cover} \) has not been explicitly studied on \( d\text{-uniform hypergraphs} \) before, but we suspect that the above results should carry over from graphs to hypergraphs as well.
4.3. Clustering

Clustering is a representative task in unsupervised machine learning that has been studied in many fields. In combinatorial optimization communities, it is often formulated as the following: Given a set $P$ of points and a set $F$ of candidate centers (also known as facilities), and a metric on $X \supseteq P \cup F$ given by the distance $\rho : X \times X \rightarrow \mathbb{R}^+ \cup \{0\}$, choose $k$ centers $C \subseteq F$ to minimize some objective function $\text{cost} := \text{cost}(P, C)$. To fully specify the problem, the choices to make are the following. Let $\rho(C, p) := \min_{c \in C} \rho(c, p)$.

- **Objective function**: Three well-studied objective functions are
  
  - $k$-MEDIAN ($\text{cost}(P, C) := \sum_{p \in P} \rho(C, p)$).
  - $k$-MEANS ($\text{cost}(P, C) := \sum_{p \in P} \rho(C, p)^2$).
  - $k$-CENTER ($\text{cost}(P, C) := \max_{p \in P} \rho(C, p)$).

- **Metric space**: The ambient metric space $X$ can be
  
  - A general metric space explicitly given by the distance $\rho : X \times X \rightarrow \mathbb{R}^+ \cup \{0\}$.
  - The Euclidean space $\mathbb{R}^d$ equipped with the $l_2$ distance.
  - Other structured metric spaces including metrics with bounded doubling dimension or bounded highway dimension.

While many previous results on clustering focused on non-parameterized polynomial time, there are at least three natural parameters one can parameterize: The number of clusters $k$, the dimension $d$ (if defined), and the approximation accuracy parameter $\epsilon$. In general metric spaces, parameterized approximation algorithms (mainly with parameter $k$) were considered very recently, but in Euclidean spaces, many previous results already give parameterized approximation algorithms with parameters $k, d$, and $\epsilon$.

4.3.1. General Metric Space

We can assume $X = P \cup F$ without loss of generality. Let $n := |X|$ and note that the distance $\rho : X \times X \rightarrow \mathbb{R}^+ \cup \{0\}$ is explicitly specified by $\Theta(n^2)$ numbers. A simple exact algorithm running in time $O(n^{k+1})$ can be achieved by enumerating all $k$ centers $c_1, \ldots, c_k \in F$ and assign each point $p$ to the closest center. In this setting, the best approximation ratios achieved by polynomial time algorithms are $2.611 + \epsilon$ for $k$-MEDIAN [155], $9 + \epsilon$ for $k$-MEANS [156], and $3$ for $k$-CENTER [157,158]. From the hardness side, it is NP-hard to approximate $k$-MEDIAN within a factor $1 + 2/e - \epsilon \approx 1.73 - \epsilon$, $k$-MEDIAN within a factor $1 + 8/e - \epsilon \approx 3.94 - \epsilon$, $k$-CENTER within a factor $3 - \epsilon$ [159].

While there are some gaps between the best algorithms and the best hardness results for $k$-MEDIAN and $k$-MEANS, it is an interesting question to ask how parameterization by $k$ changes the approximation ratios for both problems. Cohen-Addad et al. [143] studied this question and gave exact answers.

**Theorem 22 ([143]).** For any $\epsilon > 0$, there is an $(1 + 2/e + \epsilon)$-approximation algorithm for $k$-MEDIAN, and an $(1 + 8/e + \epsilon)$-approximation algorithm for $k$-MEANS, both running in time $(O(k \log k/e^2))^k n^{O(1)}$.

There exists a function $g : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ such that assuming the Gap-ETH, for any $\epsilon > 0$, any $(1 + 2/e - \epsilon)$-approximation algorithm for $k$-MEDIAN, and any $(1 + 8/e - \epsilon)$-approximation algorithm for $k$-MEANS, must run in time at least $n^{k \Omega(1)}$.

These results show that if we parameterize by $k, 1 + 2/e$ (for $k$-MEDIAN) and $1 + 8/e$ (for $k$-MEANS) are the exact limits of approximation for parameterized approximation algorithms. Similar reductions also show that no parameterized approximation algorithm can achieve $(3 - \epsilon)$-approximation for $k$-CENTER for any $\epsilon > 0$ (only assuming W[2] $\neq$ FPT), so the power of parameterized approximation is exactly revealed for all three objective functions.
Algorithm for k-MEDIAN. We briefly describe ideas for the algorithm for k-MEDIAN in Theorem 22. The main technical tool that the algorithm uses is a coreset, which will be also frequently used for Euclidean subspaces in the next subsection.

When $S$ is a set of points with weight functions $w : S \to \mathbb{R}^+$, let us extend the definition of the objective function $\text{cost}(S, C)$ such that

$$\text{cost}(S, C) := \sum_{p \in S} w(p) \cdot \rho(C, s).$$

Given a clustering instance $(P, F, \rho, k)$ and $\epsilon > 0$, a subset $S \subseteq P$ with weight functions $w : S \to \mathbb{R}^+$ is called a (strong) coreset if for any $k$ centers $C = \{c_1, \ldots, c_k\} \subseteq F$,

$$|\text{cost}(S, C) - \text{cost}(P, C)| \leq \epsilon \cdot \text{cost}(P, C).$$

For a general metric space, Chen [160] gave a coreset of cardinality $\tilde{O}(k^2 \log^2 n / \epsilon^2)$. (In this subsection, $\tilde{O}(\cdot)$ hides $\text{poly}(\log \log n, \log k, \log(1/\epsilon))$.) This was improved by Feldman and Langeberg [161] to $O(k \log n / \epsilon^2)$. We introduce high-level ideas of [160] later.

Given the coreset, it remains to give a good parameterized approximation algorithm for the problem for a much smaller (albeit weighted) point set $|P| = O(\text{poly}(\log n, k))$. Note that $|F|$ can be still as large as $n$, so naively choosing $k$ centers from $F$ will take $n^k$ time and exhaustively partitioning $P$ into $k$ sets will take $k^{|P|} = n^{\text{poly}(k)}$ time. (Indeed, exactly solving this small case will give an EPAS, which will contradict the Gap-ETH.)

Fix an optimal solution, and let $C^* = \{c^*_1, \ldots, c^*_k\}$ are the optimal centers and $P^*_i \subseteq P$ is the cluster assigned to $c^*_i$. One information we can guess is, for each $i \in [k]$, the point $p_i \in P^*_i$ closest to $c^*_i$ and (approximate) $\rho(c^*_i, p_i)$. Since $|P| = \text{poly}(k \log n)$, guessing them only takes time $(k \log n)^{O(k)}$, which can be made FPT by separately considering the case $(\log n)^k \leq n$ and the case $(\log n)^k \geq n$, in which $k = \Omega(\log n / \log \log n)$ and $(\log n)^k = (k \log k)^{O(k)}$.

Let $F_i \subseteq F$ be the set of candidate centers that are at distance approximately $r_i$ from $p_i$, so that $c^*_i \in F_i$ for each $i$. The algorithm chooses $k$ centers $C \subseteq F$ such that $|C \cap F_i| \geq 1$ for each $i \in [k]$. Let $c_i \in C \cap F_i$. For any point $p \in P$ (say $p \in P^*_i$), though the algorithm doesn’t need to know $j$, then

$$\rho(C, p) \leq \rho(c_i, p) \leq \rho(c_i, p_j) + \rho(p_j, c^*_i) + \rho(c^*_i, p) \leq 3\rho(c^*_i, p),$$

where $\rho(c_j, p_j) \approx \rho(p_j, c^*_i) \leq \rho(c^*_i, p)$ by the choice of $p_j$.

This immediately gives a 3-approximation algorithm in FPT time, which is worse than the best polynomial time approximation algorithm. To get the optimal $(1 + 2/e)$-approximation algorithm, we further reduce the job of finding $c_i \in F_i$ to maximizing a monotone submodular function with a partition matroid constraint, which is known to admit an optimal $(1 - 1/e)$-approximation algorithm [162]. Then we can ensure that for $(1 - 1/e)$ fraction of points, the distance to the chosen centers is shorter than in the optimal solution, and for the remaining $1/e$ fraction, the distance is at most three times the distance in the optimal solution. We refer the reader to [143] for further details.

Constructing a coreset. As discussed above, a coreset is a fundamental building block for optimal parameterized approximation algorithms for k-MEDIAN and k- MEANS for general metrics. We briefly describe the construction of Chen [160] that gives a coreset of cardinality $\tilde{O}(k^2 \log^2 n / \epsilon^2)$ for k-MEDIAN. Similar ideas can be also used to obtain an EPAS for Euclidean spaces parameterized by $k$, though better specific constructions are known in Euclidean spaces.

We first partition $P$ into $P_1, \ldots, P_\ell$ such that $\ell = O(k \log n)$ and

$$\sum_{i=1}^\ell |P_i| \text{diam}(P_i) = O(\text{OPT}).$$
Such a partition can be obtained by using a known (bicriteria) constant factor approximation algorithm for \(k\)-MEDITAN. Next, let \(t = O(k \log n)\), and for each \(i = 1, \ldots, \ell\), we let \(S_i = \{s_1, \ldots, s_t\}\) be a random subset of \(\ell\) points of \(P_i\) where each \(s_j\) is an independent and uniform sample from \(P_i\) and is given weight \(|P_i|/t\). (If \(|P_i| \leq t\), we simply let \(S_i = P_i\) with weights 1.) The final coreset \(S\) is the union of all \(S_i\)'s.

To prove that it works, we simply need to show that for any set of \(k\) centers \(C \subseteq F\) with \(|C| = k\),

\[
\Pr[|\text{cost}(S, C) - \text{cost}(P, C)| > \epsilon \cdot \text{cost}(P, C)] \leq o(1/n^k),
\]

so that the union bound over \(\binom{d}{k}\) choices of \(C\) works. Indeed, we show that for each \(i = 1, \ldots, \ell\),

\[
\Pr[|\text{cost}(S_i, C) - \text{cost}(P_i, C)| > \epsilon \cdot |P_i| \cdot \text{diam}(P_i)] \leq o(1/(\ell \cdot n^k)),
\]

so that we can also union bound and sum over \(i \in [\ell]\), using the fact that \(\sum_{|P_i|} \text{diam}(P_i) = O(OPT) \leq O(\text{cost}(P, C))\).

It is left to prove (1). Fix \(C\) and \(i\) (let \(P_i = \{p_1, \ldots, p_{|P_i|}\}\)), and recall that

\[
\text{cost}(P_i, C) = \sum_{j=1}^{|P_i|} \rho(C, p_j).
\]

When \(|P_i| \leq t\), \(S_i = P_i\), so (1) holds. Otherwise, recall that \(S_i = \{s_1, \ldots, s_t\}\) where each \(s_j\) is an independent and uniform sample from \(P_i\) with weight \(w := |P_i|/t\). For \(j = 1, \ldots, t\), let \(X_j := w \cdot \rho(C, s_j)\). Note that \(\text{cost}(S_i, C) = \sum_j X_j\) and \(\text{cost}(P_i, C) = t \cdot \mathbb{E}[X_j] = \mathbb{E}[\text{cost}(S_i, C)]\). A crucial observation is that that \(|\rho(C, p_j) - \rho(C, p_{j'})| \leq \text{diam}(P_i)\) for any \(j, j' \in [|P_i|]\), so that \(|X_j - X_{j'}| \leq w \cdot \text{diam}(P_i)\) for any \(j, j' \in [t]\). If we let \(X_{\min} := \min_{|X| \in [|P_i|]} (w \rho(C, p_j))\) and \(Y_j := (X_j - X_{\min})/(w \cdot \text{diam}(P_i))\), \(Y_j\)'s are \(t\) i.i.d. random variables that are supported in \([0, 1]\). The standard Chernoff–Hoeffding inequality gives

\[
\Pr \left[ |\sum_j X_j - t \mathbb{E}[X_j]| > \epsilon |P_i| \cdot \text{diam}(P_i) \right] = \Pr \left[ |\sum_j Y_j - t \mathbb{E}[Y_j]| > \epsilon t \right] \leq \exp(O(\epsilon^2 t)) \leq o(1/(\ell \cdot n^k)),
\]

proving (1) for \(t = \tilde{O}(k \log n)\) and finishing the proof. A precise version of this argument was stated in Haussler [163].

4.3.2. Euclidean Space

For Euclidean spaces, we assume that \(X = F = \mathbb{R}^d\) for some \(d \in \mathbb{N}\), endowed with the standard \(\ell_2\) metric. Let \(n = |P|\) in this subsection. Now we have \(k\) and \(d\) as natural structural parameters of clustering tasks. Many previous approximation algorithms in EUCLIDEAN \(k\)-MEDITAN and EUCLIDEAN \(k\)-MEANS in Euclidean spaces, without explicit mention to parameterized complexity, are parameterized approximation algorithms parameterized by \(k\) or \(d\) (or both). The highlight of this subsection is that for both EUCLIDEAN \(k\)-MEDITAN and EUCLIDEAN \(k\)-MEANS, an EPAS exists with only one of \(k\) and \(d\) as a parameter. Without any parameterization, both EUCLIDEAN \(k\)-MEDITAN and \(k\)-MEANS are known to be APX-hard [164,165]. We introduce these results in the chronological order, highlighting important ideas.

**EUCLIDEAN \(k\)-MEDITAN with parameter \(d\).** The first PTAS for EUCLIDEAN \(k\)-MEDITAN in Euclidean spaces with fixed \(d\) appears in Arora et al. [166]. The techniques extend Arora’s previous PTAS for the EUCLIDEAN TRAVELING SALESMAN problem in Euclidean spaces [167], first proving that there exists a near-optimal solution that interacts with a quadtree (a geometric division of \(\mathbb{R}^d\) into a hierarchy of square regions) in a restricted sense, and finally finding such a tour using dynamic programming. The running time is \(n^{O(1/\epsilon)}\) for \(d = 2\) and \(n^{\log n/\epsilon^2}d^{-2}\) for \(d > 2\). Kolliopoulous and Rao [168] improved the running time to \(2^{O((\log(1/\epsilon)/\epsilon)^{d-1})} n \log^{d+6} n\), which is an EPAS with parameter \(d\).
EUCLIDEAN $k$-MEDIAN and EUCLIDEAN $k$-MEANS with parameter $k$.

An EPAS for EUCLIDEAN $k$-MEANS even with parameters both $k$ and $d$ took longer to be discovered, and first appeared when Matoušek [169] gave an approximation scheme that runs in time $O(n^{2k^2d} \log^k n)$. After this, several improvements on the running time followed Bădoiu et al. [170], De La Vega et al. [171], Har-Peled and Mazumdar [172].

Kumar et al. [173,174] gave approximation schemes for both $k$-MEDIAN and $k$-MEANS, running in time $2^{(k/\epsilon)^{O(1)}} \cdot d^n$. This shows that an EPAS can be obtained by using only $k$ as a parameter. Using this result and and improved coresets, more improvements followed [160,161,175]. The current best runtime to get $(1 + \epsilon)$-approximation is $O(ndk + d \cdot \text{poly}(k/\epsilon) + 2^{O(k/\epsilon)})$ for $k$-MEANS [175], and $O(ndk + 2^{\text{poly}(1/\epsilon,k)})$ time for $k$-MEDIAN [161].

A crucial property of the Euclidean space that allows an EPAS with parameter $k$ (which is ruled out for general metrics by Theorem 22) is the sampling property, which says that for any set $Q \subseteq \mathbb{R}^d$ as one cluster, there is an algorithm that is given only $g(1/\epsilon)$ samples from $Q$ and outputs $h(1/\epsilon)$ candidate centers such that one of them is $\epsilon$-close to the optimal center for the entire cluster $Q$ for some functions $g, h$. (For example, for $k$-MEANS, the mean of $O(1/\epsilon)$ random samples $\epsilon$-approximates the actual mean with constant probability.) This idea leads to an $(1 + \epsilon)$-approximation algorithm running in time $|P|^{f(\epsilon,k)}$. Together even with a general coreset construction of size $\text{poly}(k, \log n, 1/\epsilon)$, one already gets an EPAS with parameter $k$. Better coresets construction are also given in Euclidean spaces. Recent developments [176–178] construct core-sets of size $\text{poly}(k, 1/\epsilon)$ (no dependence on $n$ or $d$), which is further extended to the shortest-path metric of an excluded-minor graph [179].

EUCLIDEAN $k$-MEANS with parameter $d$.

Cohen-Addad et al. [180] and Friggstad et al. [181] recently gave approximation schemes running in time $n^{f(d,\epsilon)}$ using local search techniques. These results were improved to an EPAS in [182], and also extended to doubling metrics [183].

Other metrics and $k$-CENTER. For the $k$-CENTER problem an EPAS exists when parametrizing by both $k$ and the doubling dimension [184], and also for planar graphs there is an EPAS for parameter $k$, which is implied by the EPTAS of Fox-Epstein et al. [185] (cf. [184]).

There are also parameterized approximation schemes for metric spaces with bounded highway dimension [184,186,187] and various graph width parameters [188].

Capacitated clustering and other variants. Another example where the parameterization by $k$ helps is CAPACITATED $k$-MEDIAN, where each possible center $c \in F$ has a capacity $u_c \in \mathbb{N}$ and can be assigned at most $u_c$ points. It is not known whether there exists a constant-factor approximation algorithm, and known constant factor approximation algorithms either open $(1 + \epsilon)k$ centers [188] or violate capacity constraints by an $(1 + \epsilon)$ factor [189]. Adamczyk et al. [190] gave a $(7 + \epsilon)$-approximation algorithm in $f(k, \epsilon)n^{O(1)}$ time, showing that a constant factor parameterized approximation algorithm is possible. The approximation ratio was soon improved to $(3 + \epsilon)$ [191]. For CAPACITATED EUCLIDEAN $k$-MEANS, [192] also gave a $(69 + \epsilon)$-approximation algorithm for in $f(k, \epsilon)n^{O(1)}$ time.

While the capacitated versions of clustering look much harder than their uncapacitated counterparts, there is no known theoretical separation between the capacitated version and the uncapacitated version in any clustering task. Since the power of parameterized algorithms for uncapacitated clustering is well understood, it is a natural question to understand the “capacitated VS uncapacitated question” in the FPT setting.

**Open Question 10.** Does CAPACITATED $k$-MEDIAN admit an $(1 + 2/\epsilon)$-approximation algorithm in FPT time with parameter $k$? Do CAPACITATED EUCLIDEAN $k$-MEANS / $k$-MEDIAN admit an EPAS with parameter $k$ or $d$?
Since clustering is a universal task, like capacitated versions, many variants of clustering tasks have been studied including $k$-MEDIAN/$k$-MEANS WITH OUTLIERS [193] and MATROID/KNAPSACK MEDIAN [194]. While no variant is proved to harder than the basic versions, it would be interesting to see whether they all have the same parameterized approximability with the basic versions.

4.4. Network Design

In network design, the task is to connect some set of vertices in a metric, which is often given by the shortest-path metric of an edge-weighted graph. Two very prominent problems of this type are the TRAVELLING SALESPERSON (TSP) and STEINER TREE problems. For TSP all vertices need to be connected in a closed walk (called a route), and the length of the route needs to be minimized [195]. For STEINER TREE a subset of the vertices (called terminals) is given as part of the input, and the objective is to connect all terminals by a tree of minimum weight in the metric (or graph). Both of these are fundamental problems that have been widely studied in the past, both on undirected and directed input graphs.

Undirected graphs. A well-studied parameter for STEINER TREE is the number of terminals, for which the problem has been known to be FPT since the early 1970s due to the work of Dreyfus and Wagner [196]. Their algorithm is based on dynamic programming and runs in $3^k n^{O(1)}$ time if $k$ is the number of terminals. Faster algorithms based on the same ideas with runtime $(2 + \delta)^k n^{O(1)}$ for any constant $\delta > 0$ exist [197] (here the degree of the polynomial depends on $\delta$). The unweighted STEINER TREE problem also admits a $2^k n^{O(1)}$ time algorithm [198] using a different technique based on subset convolution. Given any of these exact algorithms as a subroutine, a faster PAS can also be found [20] (cf. Section 4.7). On the other hand, no exact polynomial-sized kernel exists [130] for the STEINER TREE problem, unless $\text{NP} \subseteq \text{coNP/poly}$. Interestingly though, a PSAKS can be obtained [18].

This kernel is based on a well-known fact proved by Borchers and Du [199], which is very useful to obtain approximation algorithms for the STEINER TREE problem. It states that any Steiner tree can be covered by smaller trees containing few terminals, such that these trees do not overlap much. More formally, a full-component is a subtree of a Steiner tree, for which the leaves coincide with its terminals. For the optimum Steiner tree $T$ and any $\epsilon > 0$, there exist full-components $C_1, \ldots, C_\ell$ of $T$ such that

1. each full-component $C_i$ contains at most $2^{\lceil 1/\epsilon \rceil}$ terminals (leaves),
2. the sum of the weights of the full-components is at most $1 + \epsilon$ times the cost of $T$, and
3. taking any collection of Steiner trees $T_1, \ldots, T_\ell$, such that each tree $T_i$ connects the subset of terminals that forms the leaves of full-component $C_i$, the union $\bigcup_{i=1}^\ell T_i$ is a feasible solution to the input instance.

Not knowing the optimum Steiner tree, it is not possible to know the subsets of terminals of the full-components corresponding to the optimum. However, it is possible to compute the optimum Steiner tree for every subset of terminals of size at most $2^{\lceil 1/\epsilon \rceil}$ using an FPT algorithm for STEINER TREE. The time to compute all these solutions is $k^{O(2^{1/\epsilon})} n^{O(1)}$, using for instance the Dreyfus and Wagner [196] algorithm. Now the above three properties guarantee that the graph given by the union of all the computed Steiner trees, contains a $(1 + \epsilon)$-approximation for the input instance. In fact, the best polynomial time approximation algorithm known to date [200] uses an iterative rounding procedure to find a $\ln(4)$-approximation of the optimum solution in the union of these Steiner trees. To obtain a kernel, the union needs to be sparsified, since it may contain many Steiner vertices and also the edge weights might be very large. However, Lokshtanov et al. [18] show that the number of Steiner vertices can be reduced using standard techniques, while the edge weights can be encoded so that their space requirement is bounded in the parameter and the cost of any solution is distorted by at most a $1 + \epsilon$ factor.
Theorem 23 ([18,20]). For the STEINER TREE problem a $(1 + \varepsilon)$-approximation can be computed in $(2 + \delta)^{(1-\varepsilon/2)} n^{O(1)}$ time for any constant $\delta > 0$ (and in $2^{(1-\varepsilon/2)} n^{O(1)}$ time in the unweighted case) for any $\varepsilon > 0$, where $k$ is the number of terminals. Moreover, a $(1 + \varepsilon)$-approximate kernel of size $(k/\varepsilon)^{O(2^{1/\varepsilon})}$ can be computed in polynomial time.

A natural alternative to the number of terminals is to consider the vertices remaining in the optimum tree after removing the terminals: a folklore result states that STEINER TREE is W[2]-hard parameterized by the number of non-terminals (called Steiner vertices) in the optimum solution. At the same time, unless P = NP there is no PTAS for the problem, as it is APX-hard [201]. However an approximation scheme is obtainable when parametrizing by the number of Steiner vertices $k$ in the optimum, and also a PSAKS is obtainable under this parameterization.

To obtain both of these results, Dvořák et al. [202] devise a reduction rule that is based on the following observation: if the optimum tree contains few Steiner vertices but many terminals, then the tree must contain (1) a large component containing only terminals, or (2) a Steiner vertex that has many terminal neighbours. Intuitively, in case (2) we would like to identify a large star with terminal leaves and small cost in the current graph, while in case (1) we would like to find a cheap edge between two terminals. Note that such a single edge also is a star with terminal leaves. The reduction rule will therefore find the star with minimum weight per contained terminal, which can be done in polynomial time. This rule is applied until the number of terminals, which decreases after each use, falls below a threshold depending on the input parameter $k$ and the desired approximation ratio $1 + \varepsilon$. Once the number of terminals is bounded by a function of $k$ and $\varepsilon$, the Dreyfus and Wagner [196] algorithm can be applied on the remaining instance, or a kernel can be computed using the PSAKS of Theorem 23. It can be shown that the reduction rule does not distort the optimum solution by much as long as the threshold is large enough, which implies the following theorem.

Theorem 24 ([202]). For the STEINER TREE problem a $(1 + \varepsilon)$-approximation can be computed in $2^{O(k^2/\varepsilon^4)} n^{O(1)}$ time for any $\varepsilon > 0$, where $k$ is the number of non-terminals in the optimum solution. Moreover, a $(1 + \varepsilon)$-approximate kernel of size $(k/\varepsilon)^{2^{O(1/\varepsilon)}}$ can be computed in polynomial time.

This theorem is also generalizable to the STEINER FOREST problem, where a list of terminal pairs is given and the task is to find a minimum weight forest in the input graph connecting each pair. In this case though, the parameter has to be combined with the number of connected components of the optimum forest [202].

A variation of the STEINER FOREST problem is the SHALLOW-LIGHT STEINER NETWORK (SLSN) problem. Here a graph with both edge costs and edge lengths is given, together with a set of terminal pairs and a length threshold $L$. The task is to compute a minimum cost subgraph, which connects each terminal pair with a path of length at most $L$. For this problem a dichotomy result was shown [203] in terms of the pattern given by the terminal pairs. More precisely, the terminal pairs are interpreted as edges in a graph for which the vertices are the terminals: if $C$ is some class of graphs, then SLSN$_C$ is the SHALLOW-LIGHT STEINER NETWORK problem restricted to sets of terminal pairs that span some graph in $C$. Let $C^*$ denote the class of all stars, and $C_\lambda$ the class of graphs with at most $\lambda$ edges. The SLSN$_{C^*}$ problem is APX-hard [201], as it is a generalization of STEINER TREE (where $L = \infty$).

At the same time, both the SLSN$_{C^*}$ and SLSN$_{C_\lambda}$ problems parameterized by the number of terminals are paraNP-hard [204], since they are generalizations of the RESTRICTED SHORTEST PATH problem (where there is exactly one terminal pair). A PAS can however be obtained for both of these problems (whenever $\lambda$ is a constant), but for no other class $C$ of demand patterns [203].

Theorem 25 ([203]). For any constant $\lambda > 0$, there is an FPTAS for the SLSN$_{C_\lambda}$ problem. For the SLSN$_{C^*}$ problem a $(1 + \varepsilon)$-approximation can be computed in $4^k (n/\varepsilon)^{O(1)}$ time for any $\varepsilon > 0$, where $k$ is the number of terminal pairs. Moreover, under Gap-ETH no $(5/3 - \varepsilon)$-approximation for SLSN$_C$ can be computed in
\( f(k)n^{O(1)} \) time for any \( \varepsilon > 0 \) and computable function \( f \), whenever \( C \) is a recursively enumerable class for which \( C \not\subseteq C^* \cup C_\lambda \) for every constant \( \lambda \).

A notable special case is when all edge lengths are 1 but edge costs are arbitrary. Then SLSN\(_C^*\) is polynomial time solvable for any constant \( \lambda \), while SLSN\(_C^*\) is FPT parameterized by the number of terminals [203]. At the same time the parameterized approximation lower-bound of Theorem 25 is still valid for this case. It is not known however, whether constant approximation factors can be obtained for SLSN\(_C\) when \( C \) is a class different from \( C_\lambda \) and \( C^* \). More generally we may ask the following question.

**Open Question 11.** Given some class of graphs \( C \not\subseteq C^* \cup C_\lambda \), which approximation factor \( \alpha_C \) can be obtained in FPT time for SLSN\(_C\) parameterized by the number of terminals?

Turning to the TSP problem, a generalization of TSP introduces deadlines until which vertices need to be visited by the computed tour. A natural parameterization in this setting is the number of vertices that have deadlines. It can be shown [205] that no approximation better than 2 can be computed when using this parameter. Nevertheless, a 2.5-approximation can be computed in FPT time [205]. The algorithm will guess the order in which the vertices with deadlines are visited by the optimum solution. It then computes a 3/2-approximation for the remaining vertices using Christofides algorithm [3]. The approximation ratio follows, since the optimum tour can be thought of as two tours, of which one visits only the deadline vertices, while the other contains all remaining vertices. The approximation algorithm incurs a cost of OPT for the former, and a cost of \( \frac{3}{2} \cdot \text{OPT} \) for the latter part of the optimum tour.

**Theorem 26 ([205]).** For the DlTSP problem a 2.5-approximation can be computed in \( O(kt \cdot k) + n^{O(1)} \) time, if the number of vertices with deadlines is \( k \). Moreover, no \( (2 - \varepsilon) \)-approximation can be computed in \( f(k)n^{O(1)} \) time for any \( \varepsilon > 0 \) and computable function \( f \), unless \( P = NP \).

**Low dimensional metrics.** Just as for clustering problems, another well-studied parameter in network design is the dimension of the underlying geometric space. A typical setting is when the input is assumed to be a set of points in some \( k \)-dimensional \( \ell_p \)-metric, where distances between points \( x \) and \( y \) are given by a function \( \text{dist}(x,y) = (\sum_{i=1}^k |x_i - y_i|^p)^{1/p} \). Two prominent examples are Euclidean metrics (where \( p = 2 \)) and Manhattan metrics (where \( p = 1 \)). The dimension \( k \) of the metric space has been studied as a parameter from the parameterized approximation point-of-view avant la lettre for quite a while. It was shown [206,207] that both STEINER TREE and TSP are paraNP-hard for this parameter (since they are NP-hard even if \( k = 2 \)), and that they are APX-hard in general metrics [201,208]. However, a PAS for Euclidean metrics both for the STEINER TREE and the TSP problems were shown to exist in the seminal work of Arora [167,209]. The techniques are similar to those used for clustering, and we refer to Section 4.3.2 for an overview.

**Theorem 27 ([167]).** For the STEINER TREE and TSP problems a \( (1 + \varepsilon) \)-approximation can be computed in \( k^{O(\sqrt{k}/\varepsilon)^{k-1}} \cdot n^2 \) time for any \( \varepsilon > 0 \), if the input consists of \( n \) points in \( k \)-dimensional Euclidean space.

This result also holds for the \( t \)-MST and \( t \)-TSP problems [167], where the cheapest tree or tour, respectively, on at least \( t \) nodes needs to be found. In this case the runtime has to be multiplied by \( t \) however.

A related setting is the parameterization by the doubling dimension of the underlying metric. That is, when the parameter \( k \) is the smallest integer such that any ball in the metric can be covered by \( 2^k \) balls of half the radius. Any point set in a \( k \) dimensional \( \ell_p \)-metric has doubling dimension \( O(k) \),
and thus the latter parameter generalizes the former. For the TSP problem the above theorem can be generalized [210] to a PAS parameterized by the doubling dimension.

**Theorem 28 ([210]).** For the TSP problem a $(1 + \epsilon)$-approximation can be computed in $2^{(k/\epsilon)O(1)} n \log^2 n$ time for any $\epsilon > 0$, if the input consists of $n$ points with doubling dimension $k$.

Given that a PAS exists for STEINER TREE in the Euclidean case, it is only natural to ask whether this is also possible for low doubling metrics. Only a QPTAS is known so far [211]. Moreover, a related parameter is the highway dimension, which is used to model transportation networks. As shown by Feldmann et al. [212] the techniques of Talwar [211] for low doubling metrics can be generalized to the highway dimension to obtain a QPTAS as well. Again, it is quite plausible to assume that a PAS exists.

**Open Question 12.** Is there a PAS for STEINER TREE parameterized by the doubling dimension? Is there a PAS for either STEINER TREE or TSP parameterized by the highway dimension?

**Directed Graphs.** When considering directed input graphs (asymmetric metrics), the DIRECTED STEINER TREE problem takes as input a terminal set and a special terminal called the root. The task is to compute a directed tree of minimum weight that contains a path from each terminal to the root. In general no $f(k)$-approximation can be computed in FPT time for any computable function $f$, when the parameter $k$ is the number of Steiner vertices in the optimum solution [202]. A notable special case is the unweighted DIRECTED STEINER TREE problem, which for this parameter admits a PAS. The techniques here are the same as those used to obtain Theorem 24 for the undirected case. However, in contrast to the undirected case which admits a PASKS, no polynomial-sized $(2 - \epsilon)$-approximate kernelization exists for DIRECTED STEINER TREE [202], unless $\text{NP} \subseteq \text{coNP}/\text{poly}$. It is an intriguing question whether a 2-approximate kernel exists.

**Open Question 13.** Is there a polynomial-sized 2-approximate kernel for the unweighted DIRECTED STEINER TREE problem parameterized by the number of Steiner vertices in the optimum solution?

If the parameter is the number of terminals, the (weighted) DIRECTED STEINER TREE problem is FPT, using the same algorithm as for the undirected version [196,197]. A different variant of STEINER TREE in directed graphs is the STRONGLY CONNECTED STEINER SUBGRAPH problem, where a terminal set needs to be strongly connected in the cheapest possible way. This problem is $\text{W}[1]$-hard parameterized by the number of terminals [213], and no $O((\log^{2-\epsilon} n))$-approximation can be computed in polynomial time [214], unless $\text{NP} \subseteq \text{ZTIME}(n^{\text{polylog}(n)})$. However, a 2-approximation can be computed in FPT time [215].

The crucial observation for this algorithm is that in any strongly connected solution, fixing some terminal as the root, every terminal can be reached from the root, while at the same time the root can be reached from each terminal. Thus the optimum solution is the union of two directed trees, of which one is directed towards the root and the other is directed away from the root, and the leaves of both trees are terminals. Hence it suffices to compute two solutions to the DIRECTED STEINER TREE problem, which can be done in FPT time, to obtain a 2-approximation for STRONGLY CONNECTED STEINER SUBGRAPH. Interestingly, no better approximation is possible with this runtime [50].

**Theorem 29 ([50,215]).** For the STRONGLY CONNECTED STEINER SUBGRAPH problem a 2-approximation can be computed in $(2 + \delta)^k n^{O(1)}$ time for any constant $\delta > 0$, where $k$ is the number of terminals. Moreover, under GapETH no $(2 - \epsilon)$-approximation can be computed in $f(k)n^{O(1)}$ time for any $\epsilon > 0$ and computable function $f$. 
A generalization of both Directed Steiner Tree and Strongly Connected Steiner Subgraph is the Directed Steiner Network problem [216], for which an edge-weighted directed graph is given together with a list of ordered terminal pairs. The aim is to compute the cheapest subgraph that contains a path from $s$ to $t$ for every terminal pair $(s, t)$. If $k$ is the number of terminals, then for this problem no $k^{1/4-\omega(1)}$-approximation can be computed in $f(k)n^{O(1)}$ time [59] for any computable function $f$, under Gap-ETH. Both a PAS and a PSABS exist [50] for the special case when the input graph is planar and bidirected, i.e., every directed edge $uv$ the reverse edge $vu$ exists and has the same cost.

Similar to the PSABS for the Steiner Tree problem, these two algorithms are based on a generalization of Borchers and Du [199]. That is, Chitnis et al. [50] show that a planar solution in a bidirected graph can be covered by planar graphs with at most $2^{O(1/\varepsilon)}$ terminals each, such that the sum of their costs is at most $1 + \varepsilon$ times the cost of the solution. These covering graphs may need to contain edges that are reverse to those in the solution, but are themselves not part of the solution. For this the underlying graph needs to be bidirected. Analogous to Steiner Tree, to obtain a kernel it then suffices to compute solutions for every possible list of ordered pairs of at most $2^{O(1/\varepsilon)}$ terminals. In contrast to Steiner Tree however, there is no FPT algorithm for this. Instead, an XP algorithm with runtime $2^{O(k^{3/2} \log k)}n^{O(\sqrt{k})}$ needs to be used, which runs in polynomial time for $k \leq 2^{O(1/\varepsilon)}$ terminals with $\varepsilon$ being a constant. After taking the union of all computed solutions, the number of Steiner vertices and the encoding length of the edge weights can be reduced in a similar way as for the Steiner Tree problem. To obtain a PAS, the algorithm will guess how the planar optimum can be covered by solutions involving only small numbers of terminals. It will then compute solutions on these subsets of at most $2^{O(1/\varepsilon)}$ terminals using the same XP algorithm.

**Theorem 30** ([50]). For the Directed Steiner Network problem on planar bidirected graphs a $(1 + \varepsilon)$-approximation can be computed in $\max\{2^{k^{2^{O(1/\varepsilon)}}}, n^{2^{O(1/\varepsilon)}}\}$ time for any $\varepsilon > 0$, where $k$ is the number of terminals. Moreover, a $(1 + \varepsilon)$-approximate kernel of size $(k/\varepsilon)^{2^{O(1/\varepsilon)}}$ can be computed in polynomial time.

### 4.5. Cut Problems

Starting from Menger’s theorem and the corresponding algorithm for $s$-$t$ Cut, graph cut problems have always been at the heart of combinatorial optimization. While many natural generalizations of $s$-$t$ Cut are NP-hard, further study of these cut problems yielded beautiful techniques such as flow-cut gaps and metric embeddings in approximation algorithms [217,218], and also important separators and randomized contractions in parameterized algorithms [219–222].

#### 4.5.1. Multicut

An instance of Undirected Multicut (resp. Directed Multicut) is an undirected (resp. directed) graph $G = (V, E)$ with $k$ pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$. The goal is to remove the minimum number of edges such that there is no path from $s_i$ to $t_i$ for every $i \in [k]$. Undirected Multiway cut (resp. Directed Multiway cut) is a special case of Undirected Multicut (resp. Directed Multiway cut) where $k$ vertices are given as terminals and the goal is to make sure there is no path between any pair of terminals. They have been actively studied from both approximation and parameterized algorithms perspectives. We survey parameterized approximation algorithms for these problems with parameters $k$ and the solution size $Opt$.

**Undirected Multicut.** Undirected Multicut admits an $O(\log k)$-approximation algorithm [223] in polynomial time, and is NP-hard to approximate within any constant factor assuming the Unique Games Conjecture [224]. Directed Multiway Cut admits an 1.2965-approximation algorithm [225] in polynomial time, and is NP-hard to approximate within a factor 1.20016 [226]. Undirected Multicut (and thus Directed Multiway Cut) admits an exact algorithm parametrized by $Opt$ [219,220].
With \( k \) as a parameter, we cannot hope for an exact algorithm or an approximation scheme, since even \textsc{Undirected Multiway Cut} with 3 terminals is NP-hard to approximate within a factor \( 12/11 - \epsilon \) for any \( \epsilon > 0 \) under the Unique Games Conjecture. However, for \textsc{Undirected Multicut} with \( k \) pairs \( (s_1, t_1), \ldots, (s_k, t_k) \), one can reduce it to \( k^{O(k)} \) instances of \textsc{Undirected Multiway Cut} with at most \( 2k \) terminals, by guessing a partition of these \( s_1, t_1, \ldots, s_k, t_k \) according to the connected components containing them in the optimal solution (e.g., \( s_i \) and \( t_j \) should be always in different groups), merging the vertices in the same group into one vertex, and solving \textsc{Undirected Multiway Cut} with the merged vertices as terminals. This shows an 1.2965-approximation algorithm for \textsc{Undirected Multicut} that runs in time \( k^{O(k)} \).  

Some recent results improve or generalize this observation. For graphs with bounded genus \( g \), Cohen-Addad et al. [227] gave an EPAS running in time \( f(g, k, \epsilon) \cdot n \log n \). Chekuri and Madan [228] considered the demand graph \( H \), which is the graph formed by \( k \) edges \( (s_1, t_1), \ldots, (s_k, t_k) \). When \( t \) is the smallest integer such that \( H \) does not contain \( t \) disjoint edges as an induced subgraph, they presented a 2-approximation algorithm that runs in time \( k^{O(t)} \).

**Directed Multicut.** Generally, \textsc{Directed Multicut} is a much harder computational task than \textsc{Undirected Multicut} in terms of both approximation and parameterized algorithms. \textsc{Directed Multicut} admits a \( \min(k, \tilde{O}(n^{11/23})) \)-approximation algorithm [229]. It is NP-hard to approximate within a factor \( k - \epsilon \) for any \( \epsilon > 0 \) for fixed \( k \) [230] under the Unique Games Conjecture, or \( 2^{\Omega(\log k \cdot n)} \) for any \( \epsilon > 0 \) [231] for general \( k \). \textsc{Directed Multiway Cut} admits an \( 2 \)-approximation algorithm [232], which is tight even when \( k = 2 \) [230]. Parameterizing by OPT, \textsc{Directed Multicut} is FPT for \( k = 2 \), but \textsc{Directed Multicut} is \( W[1] \)-hard even when \( k = 4 \) [46]. \textsc{Directed Multiway Cut} on the other hand is in FPT [221].

Since it is hard to improve the trivial \( k \)-approximation algorithm even for fixed \( k \) [230], parameterizing by \( k \) does not yield a better approximation algorithm. Chitnis and Feldmann [233] gave a \( k/2 \)-approximation algorithm that runs in time \( 2^{O(k^{2+\epsilon}n)} \), and also proved that the problem under the same parameterization is still hard to approximate within a factor 59/58 with \( k = 4 \).

**Open Question 14.** What is the best approximation ratio (as a function of \( k \)) achieved by a parameterized algorithm (with parameter OPT)? Will it be close to \( O(1) \) or \( \Omega(k) \)?

### 4.5.2. Minimum Bisection and Balanced Separator

Given a graph \( G = (V, E) \), \textsc{Minimum Edge Bisection} (resp. \textsc{Minimum Vertex Bisection}) asks to remove the fewest number of edges such that the graph is partitioned into two parts \( A \) and \( B \) with \( |A| - |B| \leq 1 \). \textsc{Balanced Edge Separator} (resp. \textsc{Balanced Vertex Separator}) is a more relaxed version of the problem where the goal is to bound the size of the largest component by \( an \) for some \( 1/2 < a < 1 \). It has been actively studied from approximation algorithms, culminating in \( O(\sqrt{\log n}) \)-approximation algorithms for both \textsc{Balanced Edge Separator} and \textsc{Balanced Vertex Separator} [218,234], and an \( O(\log n) \)-approximation algorithm for \textsc{Minimum Edge Bisection} [235].

If we parameterize by the size of optimal separator \( k \), \textsc{Minimum Edge Bisection} admits an exact parameterized algorithm [222]. While \textsc{Minimum Vertex Bisection} is \( W[1] \)-hard [219], Feige and Mahdian [236] gave an algorithm that given \( 2/3 \leq k \leq 1 \) and \( \epsilon > 0 \), in time \( 2^{O(k)} \) returns an \( (\alpha + \epsilon) \) separator of size at most \( k \).

### 4.5.3. k-Cut

Given an undirected graph \( G = (V, E) \) and an integer \( k \in \mathbb{N} \), the \textsc{k-Cut} problem asks to remove the smallest number of edges such that \( G \) is partitioned into at least \( k \) non-empty connected components. The edge contraction algorithm by Karger and Stein [237] yields a randomized exact XP algorithm running in time \( O(n^{2k}) \), which was made deterministic by Thorup [238]. There were recent improvements to the running time [154,239]. There is an exact parameterized algorithm with parameter
OPT [221,240]. For general \( k \), it admits a \( (2 - \frac{2}{k}) \)-approximation algorithm [241], and is NP-hard to approximate within a factor \( (2 - \epsilon) \) for any \( \epsilon > 0 \) under the Small Set Expansion Hypothesis [74].

A simple reduction shows that \( k \)-CUT captures \((k - 1)\)-CLIQUE, so an exact FPT algorithm with parameter \( k \) is unlikely to exist. Gupta et al. [153] gave an \( (2 - \delta) \)-approximation algorithm for a small universal constant \( \delta > 0 \) that runs in time \( f(k) \cdot n^{O(1)} \). The approximation ratio was improved to 1.81 in [154], and further to 1.66 [242]. Very recently, Lokshantov et al. [243] gave a PAS that runs in time \( (k/\epsilon)^{O(k)} n^{O(1)} \), thereby (essentially) resolving the parameterized approximability of \( k \)-CUT.

4.6. \( \mathcal{F} \)-DELETION Problems

Let \( \mathcal{F} \) be a vertex-hereditary family of undirected graphs, which means that if \( G \in \mathcal{F} \) and \( H \) is a vertex-induced subgraph of \( G \), then \( H \in \mathcal{F} \) as well. \( \mathcal{F} \)-DELETION is the problem where given a graph \( G = (V,E) \), we are supposed to find \( S \subseteq V \) such that the subgraph induced by \( V \setminus S \) (denoted by \( G \setminus S \)) belongs to \( \mathcal{F} \). The goal is to minimize \( |S| \). The natural weighted version, where there is a non-negative weight \( w(v) \) for each vertex \( v \) and the goal is to minimize the sum of the weights of the vertices in \( S \), is called Weighted \( \mathcal{F} \)-DELETION.

\( \mathcal{F} \)-DELETION captures numerous combinatorial optimization problems, including VERTEX COVER (when \( \mathcal{F} \) includes all graphs with no edges), FEEDBACK VERTEX SET (when \( \mathcal{F} \) is the set of all forests), and ODD CYCLE TRANSVERSAL (when \( \mathcal{F} \) is the set of all bipartite graphs). There are a lot more interesting graph classes \( \mathcal{F} \) studied in structural and algorithmic graph theory. Some famous examples include planar graphs, perfect graphs, chordal graphs, and graphs with bounded treewidth.

In addition to beautiful structural results that give multiple equivalent characterizations, these graph classes often admit very efficient algorithms for some tasks that are believed to be hard in general graphs. Therefore, a systematic study of \( \mathcal{F} \)-DELETION for more graph classes is not only an interesting algorithmic task by itself, but also a way to obtain better algorithms for other optimization problems when the given graph \( G \) is close to a nice class \( \mathcal{F} \) (i.e., deleting few vertices from \( G \) makes it belong to \( \mathcal{F} \)). Indeed, some algorithms for INDEPENDENT SET for noisy planar/minor-free graphs discussed in Section 4.1 use an algorithm for \( \mathcal{F} \)-DELETION as a subroutine [89].

For the maximization version where the goal is to maximize \( |V \setminus S| \), a powerful but pessimistic characterization is known. Lund and Yannakakis [244] showed that whenever \( \mathcal{F} \) is vertex-hereditary and nontrivial (i.e., there are infinitely many graphs in \( \mathcal{F} \) and out of \( \mathcal{F} \)), the maximization version is hard to approximate within a factor \( 2^{\log^{1/2+\epsilon} n} \) for any \( \epsilon > 0 \). So no nontrivial \( \mathcal{F} \) is likely to admit even a polylogarithmic approximation algorithm. However, the situation is different for the minimization problem, since VERTEX COVER admits a 2-approximation algorithm, while ODD CYCLE TRANSVERSAL [245] and PERFECT DELETION [246] are NP-hard to approximate within any constant factor approximation algorithm. (The first result assumes the Unique Games Conjecture.) It indicates that a characterization of approximabilities for the minimization versions will be more complex and challenging.

There are two (closely related) frameworks to capture large graph classes.

- Choose a graph width parameter (e.g., treewidth, pathwidth, cliquewidth, rankwidth, etc.) and \( k \in \mathbb{N} \). Let \( \mathcal{F} \) be the set of graphs \( G \) with the chosen width parameter at most \( k \). The parameter of \( \mathcal{F} \)-DELETION is \( k \).
- Choose a notion of subgraph (e.g., subgraph, induced subgraph, minor, etc.) and a finite family of forbidden graphs \( \mathcal{H} \). Let \( \mathcal{F} \) be the set of graphs \( G \) that do not have any graph in \( \mathcal{H} \) as the chosen notion of subgraph. The parameter of \( \mathcal{F} \)-DELETION is \( |\mathcal{H}| := \sum_{H \in \mathcal{H}} |V(H)| \).

Many interesting classes are capture by the above frameworks. For example, to express FEEDBACK VERTEX SET, we can take \( \mathcal{F} \) to be the set of graphs with treewidth at most 1, or equivalently, the set of graphs that does not have the triangle graph \( K_3 \) as a minor. In the rest of the subsection, we introduce known results of \( \mathcal{F} \)-DELETION under the above two parameterization. Note that under these two parameterizations, the need for approximation is inherent since the simplest problem in
both frameworks, VERTEX COVER, already does not admit a polynomial-time \((2 - \varepsilon)\)-approximation algorithm under the Unique Games Conjecture.

Finally, we mention that the parameterization by the size of the optimal solution has been studied more actively from the parameterized complexity community, where many important problems are shown to be in FPT \([247-249]\).

4.6.1. Treewidth and Planar Minor Deletion

The treewidth of a graph (see Definition 1) is arguably the most well-studied graph width parameter with numerous structural and algorithmic applications. It is one of the most important concepts in the graph minor project of Robertson and Seymour. Algorithmically, Courcelle’s theorem \([250]\) states that every problem expressible in the monadic second-order logic of graphs can be solved in FPT time parameterized by treewidth. We refer the reader to the survey of Bodlaender \([251]\).

Computing treewidth is NP-hard in general \([252]\), but if we parameterize by treewidth, it can be done in FPT time \([253]\), and there is a faster constant-factor parameterized approximation algorithm \([254]\).

Let \(k \in \mathbb{N}\) be the parameter. TREEWIDTH \(k\)-DELETION (also known as TREEWIDTH \(k\)-MODULATOR in the literature) is a special case of \(\mathcal{F}\)-DELETION where \(\mathcal{F}\) is the set of all graphs with treewidth at most \(k\). Note the case \(k = 0\) yields VERTEX COVER and \(k = 1\) yields FEEDBACK VERTEX SET.

Fomin et al. \([247]\) gave a randomized \(f(k)\)-approximation algorithm that runs in \(g(k) \cdot n m\) for some computable functions \(f\) and \(g\). The approximation ratio was improved by Gupta et al. \([255]\) that gave a deterministic \(O(\log k)\)-approximation algorithm that runs in \(f(k) \cdot n^{O(1)}\) some \(f\).

This result has immediate applications to minor deletion problems. Let \(\mathcal{H}\) be a finite set of graphs, and consider \(\mathcal{H}\)-MINOR DELETION, which is a special case of \(\mathcal{F}\)-DELETION when \(\mathcal{F}\) is the set of all graphs that do not have any graph in \(\mathcal{H}\) as a minor. Its parameterized and kernelization complexity (with parameter \(\text{OPT}\)) for family \(\mathcal{H}\) has been actively studied \([247,256,257]\).

When \(\mathcal{H}\) contains a planar graph \(H\) (also known as PLANAR \(\mathcal{H}\)-DELETION in the literature), by the polynomial grid-minor theorem \([258]\), any graph \(G \in \mathcal{F}\) has treewidth at most \(k := \text{poly}(|V(H)|)\). Therefore, in order to solve \(\mathcal{H}\)-MINOR DELETION, one can first solve TREEWIDTH \(k\)-DELETION to reduce the treewidth to \(k\) and then solve \(\mathcal{H}\)-MINOR DELETION optimally using Courcelle’s theorem \([250]\). Combined with the above algorithm for TREEWIDTH \(k\)-DELETION \([255]\), this strategy yields an \(O(\log k)\)-approximation algorithm that runs in \(f(|\mathcal{H}|) \cdot n^{O(1)}\) time.

Beyond PLANAR \(\mathcal{H}\)-DELETION, there are not many results known for \(\mathcal{H}\)-MINOR DELETION. The case \(\mathcal{H} = \{K_5, K_{3,3}\}\) is called MINIMUM PLANARIZATION and was recently shown to admit an \(O(\log^{O(1)} n)\)-approximation algorithm in \(n^{O(\log n / \log \log n)}\) time \([259]\).

While the unweighted versions of TREEWIDTH \(k\)-DELETION and PLANAR \(\mathcal{H}\)-DELETION admit an approximation algorithm whose approximation ratio only depends on \(k\) not \(n\), such an algorithm is not known for WEIGHTED TREEWIDTH \(k\)-DELETION or WEIGHTED PLANAR \(\mathcal{H}\)-DELETION. Agrawal et al. \([260]\) gave a randomized \(O(\log^{1.5} n)\)-approximation algorithm and a deterministic \(O(\log^2 n)\)-approximation algorithm that run in polynomial time for fixed \(k\), i.e., the degree of the polynomial depends on \(k\). Bansal et al. \([89]\) gave an \(O(\log n \log \log n)\)-approximation algorithm for the edge deletion version. The only graphs \(\mathcal{H}\) whose weighted minor deletion problem is known to admit a constant factor approximation algorithm are single edge (WEIGHTED VERTEX COVER), triangle (WEIGHTED FEEDBACK VERTEX SET), and diamond \([261]\). For the weighted versions, no hardness beyond VERTEX COVER is known.

Open Question 15. Does WEIGHTED TREEWIDTH \(k\)-DELETION admit an \(f(k)\)-approximation algorithm with parameter \(k\) for some function \(f\)? Does TREEWIDTH \(k\)-DELETION admit a \(c\)-approximation algorithm with parameter \(k\) for some universal constant \(c\)?
Algorithms for Treewidth k-Deletion. Here we present high-level ideals of [255,260] for Treewidth k-Deletion and Weighted Treewidth k-Deletion respectively. These two algorithms share the following two important ingredients:

1. Graphs with bounded treewidth admit good separators.
2. There are good approximation algorithms to find such separators.

Given an undirected and vertex-weighted graph $G = (V, E)$ and an integer $k \in \mathbb{N}$, let (Weighted) $k$-Vertex Separator be the problem whose goal is to remove the vertices of minimum total weight so that each connected component has at most $k$ vertices. An algorithm is called an $\alpha$-bicriteria approximation algorithm if it returns a solution whose total weight is at most $\alpha \cdot \text{OPT}$ and each connected component has at most $O(k/\alpha)$ vertices from $S$.

There exists $R \subseteq V$ such that $|R| \leq k$ and each connected component of $G \setminus R$ has at most $O(k/\alpha)$ vertices from $S$. An algorithm is called an $\alpha$-bicriteria approximation algorithm if it returns a solution whose total weight is at most $\alpha \cdot \text{OPT}$ and each connected component has at most $O(k/\alpha)$ vertices from $S$.

Weighted Treewidth k-Deletion. Agrawal et al. [260] achieves an $O(\log^{1.5} n)$-approximation for Weighted Treewidth k-Deletion in time $n^{O(k)}$. It would be interesting to see whether the running time can be made FPT with parameter $k$.

The main structure of their algorithm is top-down recursive. Deleting the optimal solution $S^*$ from $G$ reduces the treewidth of $G \setminus S^*$ to $k$, so from the forest decomposition of $G \setminus S^*$, there exists a set $M^* \subseteq G \setminus S^*$ with at most $k + 1$ vertices such that each connected component of $G \setminus (M^* \cup S^*)$ has at most $2n/3$ vertices. While we do not know $S^*$, we can exhaustively try every possible $M \subseteq V$ with $|M| \leq k + 1$ and use the bicriteria approximation algorithm for Balanced Separator to find $M$ and $S$ such that (1) $|M| \leq k + 1$, (2) $w(S) \leq O(\sqrt{\log n}) \cdot \text{OPT}$, and (3) $G \setminus (M \setminus S)$ has at most $1.1 \cdot (2n/3) \leq 3n/4$ vertices.

Let $G_1, \ldots, G_t$ be the resulting connected components of $G \setminus (S \cap M)$. We solve each $G_i$ recursively to compute $S_i$ such that each $G_i \setminus S_i$ has treewidth at most $k$. The weight of $S$ was already bounded in terms of $\text{OPT}$, but the weight of $M$ was not, so we finally need to consider the graph induced by $M \cup V(G_1) \cup \cdots \cup V(G_t)$ and delete vertices of small weight to ensure small treewidth. However, this task is easy since since the treewidth of each $G_i$ is bounded by $k$ and $|M| \leq k + 1$, which bounds the treewidth of the considered graph by $2k + 1$. So we can fetch the algorithm for small treewidth graphs to solve the problem optimally. Note that the total weight of removed vertices in this recursive call is at most $(O(\sqrt{\log n}) + 1) \cdot \text{OPT}$. Since $\sum \text{OPT}(G_i) \leq \text{OPT}(G)$ and the recursion depth is at most $O(\log n)$, the total approximation ratio is $O(\log^{1.5} n)$.

Treewidth k-Deletion. Gupta et al. [255] give an $O(\log k)$-approximation algorithm that runs in time $f(k) \cdot n^{O(1)}$ for the unweighted version of Treewidth k-Deletion. The main structure of this algorithm is bottom-up iterative refinement. The algorithm maintains a feasible solution $S \subseteq V$ (we can start with $S = V$), and iteratively uses $S$ to obtain another feasible solution $S'$. If the new solution is not smaller (i.e., $|S'| \geq |S|$), then $|S| \leq O(\log k) \cdot \text{OPT}$.

Let us focus on one refinement step with the current feasible solution $S$. Let $S'$ be the optimal solution, so that $G \setminus S'$ has treewidth at most $k$. We use the following simple lemma showing the existence of a good separator of $G$ in a finer scale than before.

Lemma 2 ([[87, 255]]). Let $H$ be a graph with treewidth at most $k$, $T \subseteq V(H)$ be any subset of vertices, and $\epsilon > 0$. There exists $R \subseteq V(H)$ such that (1) $|R| \leq \epsilon |T|$ and (2) every connected component of $H \setminus R$ has at most $O(k/\epsilon)$ vertices from $T$.

Plugging $H \leftarrow G \setminus S^*$, $T \leftarrow S$ in the above lemma and letting $S' = R \cup S^*$, we can conclude that there exists $S' \subseteq V$ such that $|S'| \leq |R| + |S^*| \leq \epsilon |S| + \text{OPT}$ and each connected component of $G \setminus S'$ has at most $O(k/\epsilon)$ vertices from $S$. 

How can we find such a set $S'$ efficiently? Note that if $S = V$, then $S'$ is an $O(k/\epsilon)$-vertex separator of $G$. Lee [116] defined a generalization of $k$-VERTEX SEPARATOR called $k$-SUBSET VERTEX SEPARATOR, where the input consists of $G = (V, E)$, $S \subseteq V$, $k \in \mathbb{N}$, and the goal is to remove the smallest number of vertices so that each connected component has at most $k$ vertices from $S$, and gave an $O(\log k)$-bicriteria approximation algorithm.

Since the above lemma guarantees that $\text{Opt}$ for $O(k/\epsilon)$-SUBSET VERTEX SEPARATOR is at most $\text{Opt}$ for TREEWIDTH $k$-DELETION plus $\epsilon |S|$, applying this bicriteria approximation algorithm yields $S'$ such that $|S'| \leq O(\log k)(\text{Opt} + \epsilon |S|)$ and each connected component of $G \setminus S'$ has at most $O(k/\epsilon)$ vertices from $S$. Since $S$ is a feasible solution, it implies that the treewidth of each connected component is bounded by $O(k/\epsilon)$, so we can solve each component optimally in time $f(k/\epsilon) \cdot n^{O(1)}$. By setting $\epsilon = 0.5$, we can see the size of new solution is strictly decreased unless $|S| = O(\log k) \cdot \text{Opt}$, finishing the proof.

### 4.6.2. Subgraph Deletion

Let $H$ be a fixed pattern graph with $k$ vertices. Given a host graph $G$, deciding whether $H$ is a subgraph of $G$ (in the usual sense) is known as SUBGRAPH ISOMORPHISM, whose parameterized complexity with various parameters (e.g., $k$, $\text{tw}(H)$, $\text{genus}(G)$, etc.) was studied by Marx and Pilipczuk [263].

Guruswami and Lee [115] studied the corresponding vertex deletion problem $H$-SUBGRAPH DELETION (called $H$-TRANSVERSAL in the paper), which is a special case of $F$-DELETION where $F$ is the set of graphs that do not have $H$ as a subgraph. Note that the problem admits a simple $k$-approximation algorithm that runs in time $O(n \cdot f(n, H))$, where $f(n, H)$ denotes time to solve SUBGRAPH ISOMORPHISM with the pattern graph $H$ and a host graph with $n$ vertices. Their main hardness result states that assuming the Unique Games Conjecture, whenever $H$ is 2-vertex connected, for any $\epsilon > 0$, no polynomial time algorithm (including algorithms running in time $n^{f(k)}$ for any $f$) can achieve a $(k - \epsilon)$-approximation. (Without the UGC, they still ruled out a $(k - 1 - \epsilon)$-approximation.)

Among $H$ that are not 2-vertex-connected, there is an $O(\log k)$-approximation algorithm when $H$ is a star (in time $n^{O(1)}$) or a path (in time $f(k) n^{O(1)}$) [115,116,264]. The algorithm for $k$-path follows from the result for TREEWIDTH $k$-DELETION, because any graph without a $k$-path has treewidth at most $k$. Whenever $H$ is a tree with $k$ vertices, detecting a copy of $H$ in $G$ with $n$ vertices can be done in $2^{O(k) n^{O(1)}}$ time [265], and it is open whether there is an $O(\log k)$-approximation algorithm for $H$-SUBGRAPH DELETION in time $f(k) \cdot n^{O(1)}$.

### 4.6.3. Other Deletion Problems

**Chordal graphs.** A graph is chordal if it does not have an induced cycle of length $\geq 4$. Chordal graphs form a subclass of perfect graphs that have been actively studied. Initially motivated by efficient kernels, approximation algorithms for CHORDAL DELETION have been developed recently. The current best results are a poly(\text{Opt})-approximation [140,266] and a $O(\log^2 n)$-approximation [260].

**Edge versions.** While this subsection focused on the vertex deletion problem, there are some results on the edge deletion, edge addition, and edge modification versions. (Edge modification allows both addition and deletion.) Cao and Sandeep [267] studied MINIMUM FILL-IN, whose goal is to add the minimum number of edges to make a graph chordal. They gave new inapproximability results implying improved time lower bounds for parameterized algorithms. Giannopoulou et al. [268] gave $O(1)$-approximation algorithms for PLANAR $H$-IMMERSION DELETION parameterized by $H$. Bliznets et al. [269] considered $H$-free edge modification for a forbidden induced subgraph $H$ and give an almost complete characterization on its approximability depending on $H$.

**Directed graphs.** There is also a large body of work on parameterized algorithms for vertex deletion problems in directed graphs. While many of the known problems (including DIRECTED FEEDBACK
VERTEX SET \[270\]) admit an exact FPT algorithm, Lokshtanov et al. \[48\] studied DIRECTED ODD CYCLE TRANSVERSAL, and proved that it is W[1]-hard and is unlikely to admit an PAS under the Parameterized Inapproximability Hypothesis (or Gap-ETH). They complemented the result by showing a 2-approximation algorithm running in time \(f(OPT)\). 

4.7. Faster Algorithms and Smaller Kernels via Approximation

The focus of this section so far has been on problems for which its exact version is intractable (i.e., W[1]/W[2]-hard) and the goal is to obtain good approximations in FPT time. In this subsection, we shift our focus slightly by asking: does approximation allow us to find faster algorithms for problems already known to be in FPT?

To illustrate this, let us consider VERTEX COVER. It is of course well-known that the exact version of the problem can be solved in FPT time, with the current best running time being \(O^*(1.2738^k)\) \[271\]. The question here would be: if we are allowed to output an \((1 - \epsilon)\)-approximate solution, instead of just an exact one, can we speed up the algorithm?

To the best of our knowledge, such a question was tackled for the first time by Bourgeois et al. \[272\] and revisited quite a few times in the literature \[20,273–276\]. As one might have suspected, the answer to this question is a YES, as stated below.

**Theorem 31** \([20]\). Let \(\delta > 0\) be such that there exists an \(O^*(\delta^k)\)-time algorithm for VERTEX COVER (e.g., \(\delta = 1.2738\)). Then, for any \(\epsilon > 0\), there is an \((1 + \epsilon)\)-approximation for VERTEX COVER that runs in \(O^*(\delta^{(1-\epsilon)k})\) time.

The main idea of the algorithm is inspired by the “local ratio” method in the approximation algorithms literature (see e.g., \[277\]) and we sketch it here. The algorithm works in two stages. In the first stage, we run the greedy algorithm: as long as we have picked less than \(2k\) vertices so far and not all edges are covered, pick an uncovered edge and add both endpoints to our solution. In the second stage, we run the exact algorithm on the remaining part of the graph to find a VERTEX COVER of size \((1 - \epsilon)k\). Since the first stage runs in polynomial time, the running time of the entire algorithm is dominated by the second stage, whose running time is \(O^*(\delta^{(1-\epsilon)k})\) as desired. The correctness of the algorithm follows from the fact that, for each selected edge in the first step, the optimal solution still needs to pick at least one endpoint. As a result, the optimal solution must pick at least \(\epsilon k\) vertices with respect to the first stage (compared to \(2k\) picked by the algorithm). Thus, when the optimal solution is of size at most \(k\), there must be a solution in the second stage of size at most \((1 - \epsilon)k\), meaning that the algorithm finds such a solution and outputs a vertex cover of size \((1 + \epsilon)k\) as claimed.

The above “approximate a small fraction and brute force the rest” approach of Fellows et al. \[20\] generalizes naturally to problems beyond VERTEX COVER. Fellows et al. \[20\] formalized the method in terms of \(a\)-fidelity kernelization and apply it to several problems, including CONNECTED VERTEX COVER, \(d\)-Hitting SET and STEINER TREE. For these problems, the method gives an \((1 + \epsilon)\)-approximation algorithm that runs in time \(O^*(\delta^{(1-\Omega(\epsilon))k})\), where \(\delta > 0\) denotes a constant for which a \(O^*(\delta^k)\)-time algorithm is known for the exact version of the corresponding problem. The approach, in some form or another, is also applicable both to other parameterized problems \[278,279\] and to non-parameterized problems (e.g., \[272\]); since the latter is out-of-scope for the survey, we will not discuss the specifics here.

An intriguing question related to this line of work is whether it must be the case that the running time of \((1 + \epsilon)\)-approximation algorithms is of the form \(O^*(\delta^{(1-\Omega(\epsilon))k})\). That is, can we get a \((1 + o(1))\)-approximation for these problems in time \(O^*(\lambda^k)\) where \(\lambda\) is a constant strictly smaller than \(\delta\)? More specifically, we may ask the following:
Open Question 16. Let $\delta > 0$ be the smallest (known) constant such that an $O^*(\delta^k)$-time exact algorithm exists for VERTEX COVER. Is there an algorithm that, for any $\epsilon > 0$, runs in time $f(1/\epsilon) \cdot O^*(\lambda^k)$ for some constant $\lambda < \delta$?

Of course, the question applies not only for VERTEX COVER but other problems in the list as well. The informal crux of this question is whether, in the regime of very good approximation factors (i.e., $1 + o(1)$), approximation can still be exploited in such a way that the algorithm works significantly better than the approach “approximate a $o(1)$ fraction and then brute force”.

Turning back once again to our running example of VERTEX COVER, it turns out that algorithms faster than “approximate a small fraction and then brute force” are known [273,275,276] but only for the regime of large approximation ratios. In particular, Brankovic and Fernau [273] give faster algorithms than in Theorem 31 already for approximation ratios as small as $3/2$. The algorithms in [275,276] focus on the case of “barely non-trivial” $(2 - \rho)$-approximation factors. (Recall the greedy algorithm yields a $2$-approximation and, under the Unique Games Conjecture, the problem is NP-hard to approximate to within any constant factor less than 2.) The algorithm in [275] has a running time of $O^*(2^{k/2^{\Omega(1/\rho)}})$, which was later improved in [276] to $O^*(2^{k/2^{\Omega(1/\rho^2)}})$. These running times should be contrasted with that of “approximate a small fraction and then brute force” (i.e., applying Theorem 31 directly with $\epsilon = 1 - \rho$) which gives an algorithm with running time $O^*(2^{k\rho})$. In other words, Refs. [275,276] improve the “saving factor” from $1/\rho$ to $2^{\Omega(1/\rho)}$ and $2^{\Omega(1/\rho^2)}$ respectively. It should be noted however that, since the known $(2 - o(1))$-factor hardness of approximation is shown via the Unique Games Conjecture and unique games admit subexponential time algorithms [280,281], it is still entirely possible that this regime of approximating VERTEX COVER admits subexponential time algorithms as well. This is perhaps the biggest open question in the “barely non-trivial” approximation range:

Open Question 17. Is there an algorithm that runs in $2^{\omega(k)} n^{O(1)}$ time and achieves an approximation ratio of $(2 - \rho)$ for some absolute constant $\rho > 0$?

Let us now briefly discuss the techniques used in some of the aforementioned works. The algorithms in [273,275] are based on branching in conjunction with certain approximation techniques. (See also [282] where a similar technique is used for a related problem TOTAL VERTEX COVER.) A key idea in [273,275] is that (i) if the (average or maximum) degree of the graph is small, then good polynomial-time approximation algorithms are known [283] and (ii) if the degree is large, then branching algorithms are naturally already fast. The second part of [273] involves a delicate branching rule. However, for [275], it is quite simple: for some threshold $d$ (to be specified), as long as there exists a vertex with degree at least $d$, then (1) with some probability, simply add the vertex to the vertex cover, or (2) branch on both possibilities of it being inside the cover and outside. After this branching finishes and we are left with low-degree graphs, just run the known polynomial-time approximation algorithms [283] on these graphs. The point here is that the “error” incurred if option (1) is chosen will be absorbed by the approximation. By carefully selecting $d$ and the probability, one can arrive at the desired running time and approximation guarantee. This algorithm is randomized, but can be derandomized using the sparsification lemma [284].

To the best of our knowledge, this “barely non-trivial approximation” regime has not been studied beyond VERTEX COVER. In particular, while Bansal et al. [275] apply their techniques on several problems, these are not parameterized problems and we are not aware of any other parameterized study related to the regime discussed here.

Parallel to the running time questions we have discussed so far, one may ask an analogous question in the kernelization regime: does approximation allow us to find smaller kernels for problems that already admit polynomial-size kernels? As is the case with exact algorithms, parameterized approximation algorithms go hand in hand with approximate kernels. Indeed, many algorithmic improvements mentioned can also be viewed as improvements in terms of the size of the kernels.
In particular, recall the proof sketch of Theorem 31 for VERTEX COVER. If we stop and do not proceed with brute force in the second step, then we are left with an \((1 + \varepsilon)\)-approximate kernel. It is also not hard to argue that, by for instance applying the standard \(2k\)-size kernelization at the end, we are left with at most \(2(1 - \varepsilon)k\) vertices. This improves upon the best known \(2k - \Theta(\log k)\) bound for the exact kernel [285]. A similar improvement is known also for \(d\)-HITTING SET [20].

5. Future Directions

Although we have provided open questions along the way, we end this survey by zooming out and discussing some general future directions or meta-questions, which we find to be interesting and could be the basis for future work.

5.1. Approximation Factors

The quality of a polynomial-time approximation algorithm is mainly measured by the obtainable approximation factor \(\alpha\): the smaller it is the more feasibly solvable the problem is. Therefore, a lot of work has been invested into determining the smallest obtainable approximation factor \(\alpha\) for all kinds of computationally hard problems. In the non-parameterized (i.e., NP-hardness) world, a whole spectrum of approximability has been discovered (cf. [3,4]): the most feasibly solvable NP-hard problems (e.g., the KNAPSACK problem) admit a so-called polynomial-time approximation scheme (PTAS), which is an algorithm computing a \((1 + \varepsilon)\)-approximation for any given constant \(\varepsilon > 0\). Some problems can be shown not to admit a PTAS (under reasonable complexity assumptions), but still allow constant approximation factors (e.g., the STEINER TREE problem). Yet others can only be approximated within a polylogarithmic factor (e.g., the SET COVER problem), while some are even harder than this, as the best approximation factor obtainable is polynomial in the input size (e.g., the CLIQUE problem).

In contrast to polynomial-time approximation algorithms, a full spectrum of obtainable approximation ratios is still missing when allowing parameterized runtimes. Instead, only some scattered basic results are known. In particular, most of parameterized approximation problems belongs to one of the following categories:

- A parameterized approximation scheme (PAS) exists, i.e., for any constant \(\varepsilon > 0\) a \((1 + \varepsilon)\)-approximation can be computed in \(f(k)n^{O(1)}\) time for some parameter \(k\). These are currently the most prevalent types of results in the literature. To just mention one example, the STEINER TREE problem is APX-hard, but admits a PAS [202] when parameterized by the number of non-terminals (so-called Steiner vertices) in the optimum solution (cf. Section 4.4).
- A lower bound excluding any non-trivial approximation factor exists. For example, under ETH the DOMINATING SET problem has no \(g(k)\)-approximation in \(f(k)n^{o(k)}\) time [35] for any functions \(g\) and \(f\), where \(k\) is the size of the largest dominating set.
- A polynomial-time approximation algorithm can achieve a similar approximation ratio, i.e., the parameterization is not very helpful. For instance, for the \(k\)-CENTER problem [286] a \(2\)-approximation can be computed in polynomial time [287], but even when parameterizing by \(k\) no \((2 - \varepsilon)\)-approximation is possible [187] for any \(\varepsilon > 0\), under standard complexity assumptions. A similar situation holds for MAX \(k\)-COVERAGE, which we discussed in Section 4.2.2.
- Constant or logarithmic approximation ratios can be shown, and which beat any approximation ratio obtainable in polynomial time. For instance, STRONGLY CONNECTED STEINER SUBGRAPH problem : under standard complexity assumptions, for this problem no polynomial-time \(O(\log^{\frac{1}{\varepsilon}} n)\)-approximation algorithm exists [214], and there is no FPT algorithm parameterized by the number \(k\) of terminals [213]. However it is not hard to compute a \(2\)-approximation in \(2^{O(k)}n^{O(1)}\) time [215], and no \((2 - \varepsilon)\)-approximation algorithm with runtime \(f(k)n^{O(1)}\) exists [50] under Gap-ETH, for any function \(f\) and any \(\varepsilon > 0\) (cf. Section 4.4).

For many problems discussed in this survey, including DENSEST \(k\)-SUBGRAPH, STEINER TREE with bounded doubling/highway dimension, it has not been determined which category they belong.
There are also a lot of problems in the final category for which asymptotically tight approximation ratios have not been found, including DIRECTED MULTICUT, TREETHICKNESS k-DELETION (both weighted and unweighted). The parameterized approximability of \( \mathcal{H}\)-MINOR DELETION for non-planar \( \mathcal{H}\) is also well open except MINIMUM PLANARIZATION (\( \mathcal{H}\) = \{\( K_5, K_{3,3}\}\}) \cite{259}. It is an immediate but still interesting direction to prove tight parameterized approximation ratios for these (and more) problems.

Digressing, we remark that this survey does not include FPT-approximation of counting problems, such as approximately counting number of \( k\)-paths in a graph. The best \((1 + \epsilon)\)-multiplicative factor algorithm known \cite{288,289} for counting number of \( k\)-paths runs in time \( 4^k f(\epsilon)\) poly\( (n)\) for some subexponential function \( f\) (cf. \cite{290}). So a natural question is: can we count \( k\)-paths approximately in time \( c^k\), where \( c\) is as close to the base of running time of the algorithm of deciding existence of \( k\)-Path in a graph (the best currently known \( c\) is roughly 1.657 \cite{291,292})?

5.2. Parameterized Running Times

The quality of FPT algorithms is mainly measured in the obtainable runtime. Given a parameter \( k\), for some problems the optimum solution can be computed in \( f(k)n^{O(k)}\) time, for some functions \( f\) and \( g\) independent of the input size \( n\) (i.e., the degree of the polynomial also depends on the parameter). If such an algorithm exists the problem is slice-wise polynomial (XP), and the algorithm is called an XP algorithm. A typical example is if a solution of size \( k\) are few and far between. In particular, most of them show that for a given parameter \( k\) are constant do not even allow XP algorithms unless \( P = NP\) (e.g., the \( c\)-parameter for \( k\)-PATH on an \( \epsilon\)-parameter for \( k\)-PATH on a graph).

The discovery of the \( W\)-hierarchy in complexity theory has paved the way to providing evidence when an FPT algorithm is unlikely to exist. Assuming \( \text{ETH}\), it is even possible to provide lower bounds on the runtimes obtainable by any FPT or XP algorithm. Similar to approximation algorithms, this has lead to the discovery of a spectrum of tractability (cf. \cite{6}): starting from slightly sub-exponential \( 2^{O(\sqrt{n})}n^{O(1)}\) time, through single exponential \( 2^{O(k)}n^{O(1)}\) time, to double exponential \( 2^{O(k)}n^{O(1)}\) time for FPT algorithms with matching asymptotic lower bounds under \( \text{ETH}\) (e.g., for the PLANAR VERTEX COVER, VERTEX COVER, and EDGE CLIQUE COVER problems, respectively, each parameterized by the solution size). For XP algorithms, asymptotically tight runtime bounds of the form \( n^{O(\sqrt{n})}\) and \( n^{O(k)}\) can be obtained under \( \text{ETH}\) (e.g., for the CLIQUE parameterized by the solution size, and the PLANAR BIDIRECTED STEINER NETWORK problem parameterized by the number of terminals \cite{50}, respectively). Finally, problems that are \( NP\)-hard when the given parameter is constant do not even allow XP algorithms unless \( P = NP\) (e.g., the GRAPH COLOURING problem where the parameter is the number of colours).

In terms of tight runtime bounds, existing results on parameterized approximation algorithms are few and far between. In particular, most of them show that for a given parameter \( k\) one of the following cases applies.

- An approximation is possible in \( f(k)n^{O(1)}\) time for some function \( f\). Most current results are only concerned with the existence of an algorithm with this type of runtime, i.e., they do not provide any evidence that the obtained runtime is best possible, or try to optimize it. The only lower bounds known exclude certain types of approximation schemes when a hardness result for the parameterization by the solution size exists. For instance, it is known that if some problem does not admit a \( 2^{O(k)}n^{O(1)}\) time algorithm for this parameter \( k\) then it also does not admit an EPTAS with runtime \( 2^{O(1/k)}n^{O(1)}\) (cf. \cite{5,8}).

- A certain approximation ratio cannot be obtained in \( f(k)n^{O(1)}\) time for any function \( f\). For example, it is known that while a 2-approximation for the STRONGLY CONNECTED STEINER SUBGRAPH problem can be computed in \( 2^{O(k)}n^{O(1)}\) time \cite{215}, where \( k\) is the number of terminals, no \((2 - \epsilon)\)-approximation can be computed in \( f(k)n^{O(1)}\) time \cite{50} for any function \( f\), under \( \text{Gap-ETH}\) (cf. Section 4.4).
Hence, matching lower bounds on the time needed to compute an approximation are missing. For example, is the runtime of $2^{O(k)\cdot n^{O(1)}}$ best possible to compute a 2-approximation for the STRONGLY CONNECTED STEINER SUBGRAPH problem? Could there be a $2^{O(\sqrt{k})\cdot n^{O(1)}}$ time algorithm to compute a 2-approximation as well? For PASs the exact obtainable runtime is often elusive, even if certain types of approximation schemes can be excluded. For instance, for the STEINER TREE problem parameterized by the number of Steiner vertices in the optimum solution a $(1 + \varepsilon)$-approximation can be computed in $2^{O(k^2/\varepsilon^4)}\cdot n^{O(1)}$ time [202]. Is the dependence on $k$ and $\varepsilon$ best possible? Could there be a $2^{O(k/\varepsilon^4)}\cdot n^{O(1)}$ or $2^{O(k^2/\varepsilon^4)}\cdot n^{O(1)}$ time algorithm as well?

We remark that, for problems for which straightforward algorithms are known to be (essentially) the best possible in FPT time, or for which an improvement over polynomial time approximation is not possible, sometimes tight running time lower bounds are known in conjunction with tight inapproximability ratios. This includes $k$-DOMINATING SET (Section 3.1.2), $k$-CLIQUE (Section 3.2.1) and MAX $k$-COVERAGE (Section 4.2.2).

5.3. Kernel Sizes

The development of compositionality has lead to a theory from which lower bounds on the size of the smallest possible kernel of a problem can be derived (under reasonable complexity assumptions). The spectrum (cf. [6]) here reaches from polynomial-sized kernels (e.g., for any $q \geq 3$ and $\varepsilon > 0$ the $q$-SAT problem parameterized by the number of variables $n$ has no $O(n^{q-\varepsilon})$-sized kernel) to exponential-sized kernels (e.g., the STEINER TREE problem parameterized by the number of terminals does not admit any polynomial-sized kernel despite being FPT).

For approximate kernels, only a small number of publications exist, and the few known results fall into two categories:

- A polynomial-sized approximate kernelization scheme (PSAKS) exists, i.e., for any $\varepsilon > 0$ there is a $(1 + \varepsilon)$-approximate kernelization algorithm that computes a $(1 + \varepsilon)$-approximate kernel of size polynomial in the parameter $k$. For example, the STEINER TREE problem admits a PSAKS for both the parameterization in the number of terminals [18] and in the number of Steiner vertices in the optimum [202], even though neither of these two parameters admits a polynomial-sized (exact) kernel.

- A lower bound excluding any approximation factor for polynomial-sized kernels exists. For example, the LONGEST PATH problem parameterized by the maximum path length has no $\alpha$-approximate polynomial-sized kernel for any $\alpha$ [18], despite being FPT for this parameter [6].

Hence again the intermediate cases, for which tight constant or logarithmic approximation factors can be proved for polynomial-sized kernels, are missing. Studying approximate kernelization algorithms however is of undeniable importance to the field of parameterized approximation algorithms, as witnessed by the importance of exact kernelization to fixed-parameter tractability.

5.4. Completeness in Hardness of Approximation

A final direction we would like to highlight is to obtain more completeness in inapproximability results. Most of the results so far for FPT hardness of approximation either (i) rely on gap hypothesis or (ii) yield a hardness in terms of the W-hierarchy but the exact version of the problem is known to be complete on an even higher level (e.g., DOMINATING SET is known to be W[1]-hard to approximate but its exact version is W[2]-complete). We have discussed (i) extensively in Section 3.2 and some examples of (ii) in Section 3.1. There are also some examples of (ii) that are not covered here; for instance, Marx [293] showed W[1]-hardness for certain monotone/anti-monotone circuit satisfiability problems and the exact versions of these problems are known to be complete for higher levels of the W-hierarchy. The situation here is unlike that in the theory of NP-hardness of approximation; there the PCP Theorem [21,22] implies NP-completeness of optimization problems [294].
Thus, in the parameterized inapproximability arena, the main question here is whether we can prove completeness results for hardness of approximation for the aforementioned problems. The two important examples here are: is $k$-CLIQUE $W[1]$-hard to approximate, and is $k$-DOMINATING SET $W[2]$-hard to approximate? As discussed in Section 3.2, the former is also closely related to resolving PIH.

Finally, we note that, while completeness results are somewhat rare in FPT hardness of approximation, some are known. We give two such examples here. First is the $k$-STEINER ORIENTATION problem, discussed in Section 3.1.3; it is $W[1]$-complete to approximate [44]. Second is the MONOTONE CIRCUIT SATISFIABILITY problem (without depth bound), which was proved to be $W[P]$-complete by Marx [293]. However, it does not seem clear to us whether these techniques can be applied elsewhere, e.g., for $k$-CLIQUE.

Author Contributions: Writing—original draft and writing—review and editing, A.E.F., K.C.S., E.L. and P.M. All authors have read and agreed to the published version of the manuscript.

Funding: Andreas Emil Feldmann is supported by the Czech Science Foundation GACR (grant #19-27871X), and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004). Karthik C. S. is supported by ERC-CoG grant 772839, the Israel Science Foundation (grant number 552/16), and from the Len Blavatnik and the Blavatnik Family foundation. Euiwoong Lee is supported by the Simons Collaboration on Algorithms and Geometry.

Conflicts of Interest: The authors declare no conflict of interest.

References and Notes
10. Rubinstein, A.; Williams, V.V. SETH vs. Approximation. SIGACT News 2019, 50, 57–76. [CrossRef]
15. Kratsch, S. Polynomial kernelizations for MIN F + II, and MAX NP. Algorithmica 2012, 63, 532–550. [CrossRef]
28. Recall that there is a pair of polynomial-time L-reductions between the minimum dominating set problem and the set cover problem [27].
30. Lai, W. The Inapproximability of k-DominatingSet for Parameterized AC 0 Circuits. *Algorithms* 2019, 12, 230. [CrossRef]
33. van Emde-Boas, P. Another NP-Complete Partition Problem and the Complexity of Computing Short Vectors in a Lattice; Report Department of Mathematics; University of Amsterdam: Amsterdam, The Netherlands, 1981.
37. We could have skipped this boosting step, had we chosen a different good code with distance $\alpha$ but over a larger alphabet. For example, taking the Reed Solomon code over alphabet $\log n$ would have sufficed. We chose not to do so, to keep the proof as elementary as possible.
38. This reduction (which employs the hypercube set system) is used in [29] for proving hardness of approximating $\text{MAX } k$-$\text{COVER}$. For $\text{SET } \text{COVER}$, Feige used a more efficient set system which is not needed in our context.
43. Dinur, I. The PCP theorem by gap amplification. *J. ACM* 2007, 54, 12. [CrossRef]
47. We remark that the original conjecture in [48] says that the problem is W[1]-hard to approximate. However, we choose to state the more relaxed form here.
56. See also the related Projection Game Conjecture (PGC) [61].
57. Naturally, we say that two functions \( f_i \) and \( f_j \) agree iff \( f_i(x) = f_j(x) \) for all \( x \in S_i \cap S_j \).
64. Raz, R.; Safra, S. A Sub-Constant Error-Probability Low-Degree Test, and a Sub-Constant Error-Probability PCP Characterization of NP. In Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, TX, USA, 4–6 May 1997; pp. 475–484. [CrossRef]
69. Chan, S.O. Approximation Resistance from Pairwise-Independent Subgroups. J. ACM 2016, 63, 27. [CrossRef]
74. Manurangsi, P. Inapproximability of maximum biclique problems, minimum k-cut and densest at-least-k-subgraph from the small set expansion hypothesis. Algorithms 2018, 11, 10. [CrossRef]
75. We note, however, that strong inapproximability of BICLIQUE is known under stronger assumptions [72–74].
79. Again, similar to BICLIQUE, DENSEST k-SUBGRAPH is known to be hard to approximate under stronger assumptions [72,76–78].
100. Alber, J.; Fiala, J. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. J. Algorithms 2004, 52, 134–151. [CrossRef]
103. Sometimes called IMPROPER COLORING.


109. In [105] the runtime of these algorithms is stated as \((\log n / \varepsilon)^{O(k)} 2^k n^{O(1)}\), which can be shown to be upper bounded by \((k / \varepsilon)^{O(k)} n^{O(1)}\) (see e.g., ([108] Lemma 1)).


133. Recall that a bi-kernel is similar to a kernel except that its the output need not be an instance of the original problem. Bi-PSAKS can be defined analogously to PSAKS, but with bi-kernel instead of kernel. In the case of CONNECTED DOMINATING SET, the bi-kernel outputs an instance of an annotated variant of CONNECTED DOMINATING SET, where some vertices are marked and do not need to be covered by the solution.


137. See Definition 1 for the definition of the treewidth.


145. The problem has also been referred to as MIN k-UNION and SMALL SET BIPARTITE VERTEX EXPANSION in the literature [150,151].

146. The argument of [146] was later independently rediscovered in [147] as well.


158. A special case that has received significant attention assumes $P = F$. In this case, the best approximation ratio for k-CENTER becomes 2.


167. Arora, S. Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM (JACM)* **1998**, **45**, 753–782. [CrossRef]


194. Swamy, C. Improved approximation algorithms for matroid and knapsack median problems and applications. 
   ACM Trans. Algorithms (TALG) 2016, 12, 49. [CrossRef]

195. Sometimes also the non-metric version of TSP is considered, which however is much harder than the metric one. We only consider the metric version here.

196. Dreyfus, S.E.; Wagner, R.A. The Steiner problem in graphs. Networks 1971, 1, 195–207. [CrossRef]


198. Nederlof, J. Fast Polynomial-Space Algorithms Using Möbius Inversion: Improving on Steiner Tree and 
   Related Problems. In Proceedings of the Automata, Languages and Programming, 36th International 


200. Byrka, J.; Grandoni, F.; Rothvoss, T.; Sanità, L. Steiner Tree Approximation via Iterative Randomized 
   Rounding. J. ACM 2013, 60, 1–33. [CrossRef]

   Sci. 2008, 406, 207–214. [CrossRef]

   Schemes for Steiner Trees with Small Number of Steiner Vertices. In Proceedings of the 35th Symposium on 
   [CrossRef]

203. Babay, A.; Dinitz, M.; Zhang, Z. Characterizing Demand Graphs for (Fixed-Parameter) Shallow-Light Steiner 
   Trees. JCSS 2015, 81, 1665–1677. [CrossRef]

   [CrossRef]

   4, 237–244. [CrossRef]

   Appl. Math. 1977, 32, 835–859. [CrossRef]

   [CrossRef]


211. Feldmann, A.E.; Fung, W.S.; Könemann, J.; Post, I. A $(1+\varepsilon)$-Embedding of Low Highway Dimension 

   SIAM J. Discret. Math. 2011, 25, 583–599. [CrossRef]


   In Proceedings of the Parameterized and Exact Computation—8th International Symposium, IPEC, Sophia 

215. Sometimes also called DIRECTED STEINER FOREST; note however that the optimum is not necessarily a forest.

216. Leighton, T.; Rao, S. Multicommodity max-flow min-cut theorems and their use in designing approximation 
   algorithms. J. ACM (JACM) 1999, 46, 787–832. [CrossRef]

   (JACM) 2009, 56, 5. [CrossRef]


286. Here we consider the version where the set of candidate centers is not separately given.


294. To be more precise, these problems need to be phrased as promise problems and NP-hardness is with respect to these. We will not go into details here.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).
The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems

Andreas Emil Feldmann and Dániel Marx

1 Department of Applied Mathematics, Charles University, Prague, Czech Republic; and Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
feldmann.a.e@gmail.com
2 Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI), Budapest, Hungary
dmarx@cs.bme.hu

Abstract

Given a directed graph $G$ and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of terminal pairs, the Directed Steiner Network problem asks for a minimum-cost subgraph of $G$ that contains a directed $s_i \rightarrow t_i$ path for every $1 \leq i \leq k$. The special case Directed Steiner Tree (when we ask for paths from a root $r$ to terminals $t_1, \ldots, t_k$) is known to be fixed-parameter tractable parameterized by the number of terminals, while the special case Strongly Connected Steiner Subgraph (when we ask for a path from every $t_i$ to every other $t_j$) is known to be W[1]-hard parameterized by the number of terminals. We systematically explore the complexity landscape of directed Steiner problems to fully understand which other special cases are FPT or W[1]-hard. Formally, if $H$ is a class of directed graphs, then we look at the special case of Directed Steiner Network where the list $(s_1, t_1), \ldots, (s_k, t_k)$ of requests form a directed graph that is a member of $H$. Our main result is a complete characterization of the classes $H$ resulting in fixed-parameter tractable special cases: we show that if every pattern in $H$ has the combinatorial property of being “transitively equivalent to a bounded-length caterpillar with a bounded number of extra edges,” then the problem is FPT, and it is W[1]-hard for every recursively enumerable $H$ not having this property. This complete dichotomy unifies and generalizes the known results showing that Directed Steiner Tree is FPT [Dreyfus and Wagner, Networks 1971], Strongly Connected Steiner Subgraph is W[1]-hard [Guo et al., SIAM J. Discrete Math. 2011], and Directed Steiner Network is solvable in polynomial-time for constant number of terminals [Feldman and Ruhl, SIAM J. Comput. 2006], and moreover reveals a large continent of tractable cases that were not known before.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.2 Graph Theory

Keywords and phrases Directed Steiner Tree, Directed Steiner Network, fixed-parameter tractability, dichotomy

Digital Object Identifier 10.4230/LIPIcs.ICALP.2016.27

1 Introduction

Steiner Tree is a basic and well-studied problem of combinatorial optimization: given an edge-weighted undirected graph $G$ and a set $R \subseteq V(G)$ of terminals, it asks for a minimum-
cost tree connecting the terminals. The problem is well known to be NP-hard, in fact, it was one of the 21 NP-hard problems identified by Karp’s seminal paper [22]. There is a large literature on approximation algorithms for \textsc{Steiner Tree} and its variants, resulting for example in constant-factor approximation algorithms for general graphs and approximation schemes for planar graphs (see [8, 15, 9, 4, 3, 2, 7, 26, 24, 23, 1, 17]). From the viewpoint of parameterized algorithms, the first result is the classic dynamic-programming algorithm of Dreyfus and Wagner [17] from 1971, which solves the problem with \( k = |R| \) terminals in time \( 3^k \cdot n^{O(1)} \), showing that the problem is fixed-parameter tractable (FPT) parameterized by the number of terminals. More recently, the running time was improved to \( 2^k \cdot n^{O(1)} \) by Björklund et al. [5] using the technique of fast subset convolution. \textsc{Steiner Forest} is the generalization where the input contains an edge-weighted graph \( G \) and a list \((s_1, t_1), \ldots, (s_k, t_k)\) of pairs of terminals and the task is to find a minimum-cost subgraph containing an \( s_i \rightarrow t_i \) path for every \( 1 \leq i \leq k \). The fixed-parameter tractability of \textsc{Steiner Forest} follows from the observation that the connected components of the solution induces a partition on the set \( \{s_1, \ldots, s_k, t_1, \ldots, t_k\} \) of terminals, and hence we can solve the problem by for example trying every partition and invoking a \textsc{Steiner Tree} algorithm for each class of the partition.

On directed graphs, Steiner problems can become significantly harder, and while there is a richer landscape of variants, very few results are known [21, 11, 18, 10, 27, 14, 13]. A natural and well-studied generalization of \textsc{Steiner Tree} to directed graphs is \textsc{Directed Steiner Tree} (DST), where an arc-weighted directed graph \( G \) and terminals \( r, t_1, \ldots, t_k \) are given and the tasks is to find a minimum-cost subgraph containing an \( r \rightarrow t_i \) path for every \( 1 \leq i \leq k \). Using essentially the same techniques as in the undirected case [5, 17], one can show that this problem is also FPT parameterized by the number of terminals. An equally natural generalization of \textsc{Steiner Tree} to directed graphs is the \textsc{Strongly Connected Steiner Subgraph} (SCSS) problem, where an arc-weighted directed graph \( G \) with terminals \( t_1, \ldots, t_k \) is given, and the task is to find a minimum-cost subgraph containing a \( t_i \rightarrow t_j \) path for any \( 1 \leq i, j \leq k \) with \( i \neq j \). Guo et al. [21] showed that, unlike DST, the SCSS problem is \( \text{W}[1] \)-hard parameterized by the number \( k \) of terminals (see also [14]). A common generalization of DST and SCSS is the \textsc{Directed Steiner Network} (DSN) problem (also called \textsc{Directed Steiner Forest} or \textsc{Point-to-Point Connection}), where an arc-weighted directed graph \( G \) and a list \((s_1, t_1), \ldots, (s_k, t_k)\) of terminal pairs are given and the task is to find a minimum-cost subgraph containing an \( s_i \rightarrow t_i \) path for every \( 1 \leq i \leq k \). Being a generalization of SCSS, the \textsc{Directed Steiner Network} problem is also \( \text{W}[1] \)-hard, but Feldman\(^1\) and Ruhl [18] showed that the problem is solvable in time \( n^{O(k)} \), that is, in polynomial time for every constant \( k \).

Besides \textsc{Directed Steiner Tree}, what other special cases of \textsc{Directed Steiner Network} are fixed-parameter tractable? Our main result gives a complete map of the complexity landscape of directed Steiner problems, precisely describing all the \( \text{FPT}/\text{W}[1] \)-hard variants and revealing highly non-trivial generalizations of \textsc{Directed Steiner Tree} that are still tractable. Our results are expressed in the following formal framework. The pairs \((s_1, t_1), \ldots, (s_k, t_k)\) in the input of DSN can be interpreted as a directed (unweighted) \textit{pattern graph} on a set \( R \) of terminals. If this pattern graph is an out-star, then the problem is precisely DST; if it is a bidirected clique, then the problem is precisely SCSS. More generally, if \( \mathcal{H} \) is any class of graphs, then we define the \textsc{Directed Steiner} \( \mathcal{H} \)-\textit{Network} (\( \mathcal{H} \)-DSN) problem as the restriction of DSN where the pattern graph is a member of \( \mathcal{H} \). That is, the

\(^1\) We note that Jon Feldman (co-author of [18]) is not the same person as Andreas Emil Feldmann (co-author of this paper).
input of $\mathcal{H}$-DSN is an arc-weighted directed graph $G$, a set $R \subseteq V(G)$ of terminals, and an unweighted directed graph $H \in \mathcal{H}$ on $R$; the task is to find a minimum-cost network $N \subseteq G$ such that $N$ contains an $s \to t$ path for every $st \in E(H)$.

We give a complete characterization of the classes $\mathcal{H}$ for which $\mathcal{H}$-DSN is FPT or $W[1]$-hard. We need the following definition of “almost-caterpillar graphs” to describe the borderline between the easy and hard cases (see Figure 1).

**Definition 1.** A $\lambda_0$-caterpillar graph is constructed as follows. Take a directed path $(v_1, \ldots, v_{\lambda_0})$ from $v_1$ to $v_{\lambda_0}$, and let $W_1, \ldots, W_{\lambda_0}$ be pairwise disjoint vertex sets such that $v_i \in W_i$ for each $i \in \{1, \ldots, \lambda_0\}$. Now add edges such that either every $W_i$ forms an out-star with root $v_i$, or every $W_i$ forms an in-star with root $v_i$. In the former case we also refer to the resulting $\lambda_0$-caterpillar as an *out-caterpillar*, and in the latter as an *in-caterpillar*. A 0-caterpillar is the empty graph. The class $\mathcal{C}_{\lambda, \delta}$ contains all directed graphs $H$ such that there is a set of edges $F \subseteq E(H)$ of size at most $\delta$ for which the remaining edges $E(H) \setminus F$ span a $\lambda_0$-caterpillar for some $\lambda \leq \lambda_0$.

If there is an $s \to t$ path in the pattern graph $H$ for two terminals $s, t \in R$, then adding the edge $st$ to $H$ does not change the problem: connectivity from $s$ to $t$ is already implied by $H$, hence adding this edge does not change the feasible solutions. That is, adding a transitive edge does not change the solution space and hence it is really only the transitive closure of the pattern $H$ that matters. We say that two pattern graphs are *transitively equivalent* if their transitive closures are isomorphic. We denote the class of patterns that are transitively equivalent to some pattern of $\mathcal{C}_{\lambda, \delta}$ by $\mathcal{C}^*_{\lambda, \delta}$. Our main result is a sharp dichotomy saying that $\mathcal{H}$-DSN is FPT if every pattern of $\mathcal{H}$ is transitively equivalent to an almost-caterpillar graph and it is $W[1]$-hard otherwise. We measure the running time in $\lambda$, $\delta$, and the *vertex cover number* $\tau$ of the input pattern $H$, i.e. $\tau$ is the size of the smallest vertex subset $W$ of $H$ such that every edge of $H$ is incident to a vertex of $W$.

**Theorem 2.** Let $\mathcal{H}$ be a recursively enumerable class of patterns.

1. If there are constants $\lambda$ and $\delta$ such that $\mathcal{H} \subseteq \mathcal{C}^*_{\lambda, \delta}$, then $\mathcal{H}$-DSN with parameter $k = |R|$ is FPT and can be solved in $2^{O(k + \max(\omega^2, \tau \omega \log \omega))} n^{O(\omega)}$ time, where $\omega = (1 + \lambda)(\lambda + \delta)$ and $\tau$ is the vertex cover number of the given input pattern $H \in \mathcal{H}$.

2. Otherwise, if there are no such constants $\lambda$ and $\delta$, then the problem is $W[1]$-hard for parameter $k$.

Invoking Theorem 2 with specific classes $\mathcal{H}$, we can obtain algorithmic or hardness results for specific problems. For example, we may easily recover the following facts:

- If $\mathcal{H}_{\text{DST}}$ is the class of all out-stars, then $\mathcal{H}_{\text{DST}}$-DSN is precisely the DST problem. As $\mathcal{H}_{\text{DST}} \subseteq \mathcal{C}^*_{1,0}$ holds, Theorem 2(1) recovers the fact that DST can be solved in time $2^{|R|} n^{O(1)}$ and is hence FPT parameterized by the number $k = |R|$ of terminals [17, 5].

- If $\mathcal{H}_{\text{SCSS}}$ is the class of all bidirected cliques, then $\mathcal{H}_{\text{SCSS}}$-DSN is precisely the SCSS problem. One can observe that $\mathcal{H}_{\text{SCSS}}$ is not contained in $\mathcal{C}^*_{\lambda, \delta}$ for any constants $\lambda, \delta$ (for example, because every graph in $\mathcal{C}_{\lambda, \delta}$ has at most $\lambda + 2\delta$ vertices with both positive
in-degree and positive out-degree, and this remains true also for the graphs in $C_{\lambda,\delta}$. Hence Theorem 2(2) recovers the fact that SCSS is W[1]-hard [21].

Let $\mathcal{H}_d$ be the class of directed graphs with at most $d$ edges. As $\mathcal{H}_d \subseteq C_{\alpha_d}$ holds, Theorem 2(1) recovers the fact that Directed Steiner Network with at most $d$ requests is polynomial-time solvable for every constant $d$ [18]. Note that any pattern of $\mathcal{H}_{\text{SCSS}}$ is transitively equivalent to a bidirected star, which has vertex cover number $\tau = 1$. Hence for the important spacial case of SCSS, our algorithm recovers the running time of $2^{O(d \log d)} n^{O(d)} = n^{O(d)}$ given in [18].

Very recently, Suchý [25] studied the following generalization of DST and SCSS: in the $q$-Root Steiner Tree ($q$-RST) problem, a set of $q$ roots and a set of $k$ leaves are given, and the task is to find a minimum-cost network where the roots are in the same strongly connected component and every leaf can be reached from every root. Building on the work of [18], Suchý [25] presented an algorithm with running time $2^{O(k)} \cdot n^{O(q)}$ for this problem, which shows that it is FPT for every constant $q$. Let $\mathcal{H}_{q\text{-RST}}$ be the class of directed graphs that are obtained from an out-star by making $q - 1$ of the edges bidirected. Observe that $\mathcal{H}_{q\text{-RST}}$ is a subset of $C_{1,q-1}$, that $q$-RST can be expressed by an instance of $\mathcal{H}_{q\text{-RST}}$-DSN, and that any pattern of $\mathcal{H}_{q\text{-RST}}$ has vertex cover number $\tau = 1$. Thus Theorem 2(1) implies that $q$-RST can be solved in time $2^{O(k + q \log q)} \cdot n^{O(q)} = 2^{O(k)} \cdot n^{O(q)}$, recovering the fact that it is FPT for every constant $q$.

Thus the algorithmic side of Theorem 2 unifies and generalizes three algorithmic results: the fixed-parameter tractability of DST (which is based on dynamic programming on the tree structure of the solution) and $q$-RST (which is based on simulating a “pebble game”), and also the polynomial-time solvability of DSN with constant number of requests (which also is based on simulating a “pebble game”). Let us point out that our algorithmic results are significantly more general than just the unification of these three results: the generalization from stars to bounded-length caterpillars is already a significant extension and very different from earlier results. We consider it a major success of the systematic investigation that, besides finding the unifying algorithmic ideas generalizing all previous results, we were able to find tractable special cases in an unexpected new direction.

There is a surprising non-monotonicity in the classification result of Theorem 2. As DST is FPT and SCSS is W[1]-hard, one could perhaps expect that $\mathcal{H}$-DSN becomes harder as the pattern become denser. However, it is possible that the addition of further requests makes the problem easier. For example, if $\mathcal{H}$ contains every graph that is the vertex-disjoint union of two out-stars, then $\mathcal{H}$-DSN is classified to be W[1]-hard by Theorem 2(2). However, if we consider those graphs where there is also a directed edge from the center of one star to the other star, then these graphs are 2-caterpillars (i.e., contained in $C_{2,0}$) and hence $\mathcal{H}$-DSN becomes FPT by Theorem 2(1). This unexpected non-monotonicity further underlines the importance of completely mapping the complexity landscape of the problem area: without complete classification, it would be very hard to predict what other tractable/intractable special cases exist.

We mention that one can also study the vertex-weighted version of the problem, where the input graph has weights on the vertices and the goal is to minimize the total vertex-weight of the solution. In general, vertex-weighted problems can be more challenging than edge-weighted variants [15, 4, 23, 12]. However, for general directed graphs, there are easy transformations between the two variants. Thus the results of this paper can be interpreted for the vertex-weighted version as well.
1.1 Our techniques

We prove Theorem 2 the following way. In Section 2, we first establish the combinatorial bound that there is a solution whose cutwidth, and hence also (undirected) treewidth, is bounded by the number of requests.

→ Theorem 3. A minimal solution $M$ to a pattern $H$ has cutwidth at most $7m$ if $m = |E(H)|$.

Then in Section 3 we go on to generalize this to almost-caterpillars, showing that if the pattern is in $\mathcal{C}_{\lambda,\delta}^*$, then the (undirected) treewidth can be bounded in $\lambda$ and $\delta$.

→ Theorem 4. The treewidth of a minimal solution to any pattern graph in $\mathcal{C}_{\lambda,\delta}^*$ is at most $7(1 + \lambda)(\lambda + \delta)$.

This combinatorial bound can be exploited in an algorithm that restricts the search for a bounded-treewidth solution.

→ Theorem 5. Let an instance of $\mathcal{H}$-DSN be given by a graph $G$ with $n$ vertices, and a pattern $H$ on $k$ terminals with vertex cover number $\tau$. If the optimum solution to $H$ in $G$ has treewidth $\omega$ then the optimum can be computed in time $2^{O(k + \max(\omega^2, \tau \omega \log \omega))} n^{O(\omega)}$.

Combining Theorem 4 and Theorem 5 proves the algorithmic side of Theorem 2. We remark that the proof is completely self-contained (with the exception of some basic facts on treewidth) and in particular we do not build on the algorithms of Feldman and Ruhl [18]. As combining Theorem 3 and Theorem 5 already proves that DSN with a constant number of requests can be solved in polynomial time, as a by-product this gives an independent proof for the result of Feldman and Ruhl [18]. One can argue which algorithm is simpler, but perhaps our proof (with a clean split of a combinatorial and an algorithmic statement) is more methodological and better reveals the underlying reason why the problem is tractable.

Finally, in Section 4 we show that whenever the patterns in $\mathcal{H}$ are not transitively equivalent to almost-caterpillars, the problem is $\text{W}[1]$-hard. We first show that there is only a small number of obstacles for not being transitively equivalent to almost-caterpillars: the graph class contains (possibly after identification of vertices) arbitrarily large strongly connected graphs, pure diamonds, or flawed diamonds (see Lemma 22 for the precise statement). We provide a separate $\text{W}[1]$-hardness proof for each of these cases, completing the proof of the hardness side of Theorem 2.

Due to space limitations we defer all missing proofs to the full version of this extended abstract, including the algorithm that implies Theorem 5.

2 The cutwidth of minimal solutions for bounded-size patterns

Consider a minimal solution $M$ to an instance of $\mathcal{H}$-DSN, in which no edge can be removed without making the solution infeasible. The goal of this section is to prove Theorem 3: we bound the cutwidth of a minimal solution $M$ to a pattern $H$ in terms of $m = |E(H)|$. A layout of a graph $G$ is an injective function $\psi : V(G) \to \mathbb{N}$ inducing a total order on the vertices of $G$. Given a layout, we define the set $V_i = \{v \in V(G) \mid \psi(v) \leq i\}$ and say that an edge crosses the cut $(V_i, \overline{V}_i)$ if it has one endpoint in $V_i$ and one endpoint in $\overline{V}_i := V(G) \setminus V_i$. The cutwidth of the layout is the maximum number of edges crossing any cut $(V_i, \overline{V}_i)$ for any $i \in \mathbb{N}$. The cutwidth of a graph is the minimum cutwidth over all its layouts.

Like Feldman and Ruhl [18], we consider the two extreme cases of directed acyclic graphs (DAGs) and strongly connected components (SCCs) in our proof. Contracting all SCCs of $M$ without removing parallel edges sharing the same head and tail, but removing
the resulting self-loops, results in a directed acyclic multi-graph \( D \), the so-called condensation graph of \( M \). We bound the cutwidth of \( D \) and the SCCs of \( M \) separately, and then put together these two bounds to obtain a bound for the cutwidth of \( M \). As we will see, bounding the cutwidth of the acyclic multi-graph \( D \) and putting together the bounds are fairly simple. The main technical part is bounding the cutwidth of the SCCs.

We will need two simple facts about cutwidth. First, the cutwidth of an acyclic multi-graph can be bounded using the existence of a topological ordering of the vertices. That is, for any acyclic graph \( G \) there is an injective function \( \varphi : V(G) \to \mathbb{N} \) such that \( \varphi(u) < \varphi(v) \) if \( uv \in E(G) \). Note that such a function in particular is a layout.

\[ \text{Lemma 6. The layout given by a topological ordering } \varphi_D \text{ of an acyclic directed multi-graph } \text{that is the union of } m \text{ paths, has cutwidth at most } m. \]

\[ \text{Lemma 7. Let } G \text{ be a directed graph and } D \text{ be its condensation multi-graph. If the cutwidth of } D \text{ is } x \text{ and the cutwidth of every SCC of } G \text{ is at most } y, \text{ then the cutwidth of } G \text{ is at most } x + y. \]

\[ \text{Lemma 8. Any SCC } U \text{ of a minimal solution } M \text{ to a pattern } H \text{ with at most } m \text{ edges has cutwidth at most } 6m. \]

\[ \text{Proof. First we establish that } U \text{ is a minimal solution to a certain pattern.} \]

\[ \text{Claim 9. } U \text{ is a minimal solution to a pattern } H_U \text{ with at most } m \text{ edges.} \]

Let \( R_U \) be the terminals in the pattern \( H_U \) given by Claim 9 and let us select an arbitrary root \( t \in R_U \). Note that \( H_U \) has at most \( m \) edges, hence \( |R_U| \leq 2m \). Let \( S_{in} \) (resp., \( S_{out} \)) be an in-star (resp., out-star) connecting \( t \) with every other vertex of \( R_U \). As \( U \) is a strongly connected graph containing every vertex of \( R_U \), it is also a solution to the pattern \( S_{in} \) on \( R_U \). Let us select an \( A_{in} \subseteq U \) that is a minimal solution to \( S_{in} \); it is not hard to see that \( A_{in} \) is an in-arborescence with at most \( 2m \) leaves. Similarly, let \( A_{out} \subseteq U \) be an out-arborescence that is a minimal solution to \( S_{out} \). Observe that \( U \) has to be exactly \( A_{in} \cup A_{out} \): if there is an edge \( e \in E(U) \) that is not in \( A_{in} \cup A_{out} \), then \( U \setminus e \) still contains a path from every vertex of \( R_U \) to every other vertex of \( R_U \) though \( t \), contradicting the fact that \( U \) is a minimal solution to pattern \( H_U \).

Let \( Z \) be the set of edges obtained by reversing the edges in \( E(A_{in}) \setminus E(A_{out}) \). As reversing edges does not change the cutwidth, bounding the cutwidth of \( A_{out} \cup Z \) will also imply a bound on the cutwidth of \( U = A_{in} \cup A_{out} \).

\[ \text{Claim 10. The union } A_{out} \cup Z \text{ is a directed acyclic graph.} \]

Claim 10 implies a topological ordering on the vertices of \( A_{out} \cup Z \). This order can be used as a layout for \( U \). Using some more structural insights, the number of edges crossing a given cut can be bounded in the number of edges of the pattern graph, as the following claim shows.

\[ \text{Claim 11. Any topological ordering } \varphi \text{ of the graph } A_{out} \cup Z \text{ has cutwidth at most } 6m. \]

As the underlying undirected graph of \( U \) and \( A_{out} \cup Z \) are the same, Claim 11 implies that the cutwidth of \( U \) is at most \( 6m \). This completes the proof of Lemma 8.

The proof of Theorem 3 follows easily from putting together the ingredients. We remark that the bound on the cutwidth in Claim 11 is asymptotically tight: Take a constant degree expander on \( m \) vertices. It has treewidth \( \Omega(m) \) [20], and so its cutwidth is at least as large. Now bi-direct each (undirected) edge \( \{u, v\} \) by replacing it with the directed edges \( uv \) and \( vu \).
Next subdivide every edge $uv$ to obtain edges $ut$ and $tv$ for a new vertex $t$, and make $t$ a terminal of $R$. This yields a strongly connected instance $G$. The pattern graph $H$ for this instance is a cycle on $R$, which has $O(m)$ edges, since the terminals are subdivision points of bi-directed edges of a constant degree graph with $m$ vertices. As $H$ is strongly connected, every minimal solution to $H$ contains the edges $ut$ and $tv$ incident to each terminal $t$. Thus a minimal solution contains all of $G$ and has cutwidth $\Omega(m)$. Since $G$ is strongly connected, it also contains the required arborescences $A_{in}$ and $A_{out}$.

3 The treewidth of minimal solutions to almost-caterpillar patterns

In this section, we prove that any minimal solution $M$ to a pattern $H \in C_{1,\delta}$ has the following structure.

**Theorem 12.** A minimal solution $M$ to a pattern $H \in C_{1,\delta}$ consists of a subgraph $M^c$ that is a minimal solution to a sub-pattern $H^c$ of $H$ with at most $(1+\lambda)(\lambda+\delta)$ edges, and a forest $M \setminus M^c$ of out-arborescences, each of which intersects $M^c$ only at the root.

According to Theorem 3, the cutwidth of the core $M^c$ is therefore at most $7(1+\lambda)(\lambda+\delta)$. It is well known [6] that the cutwidth is an upper bound on the treewidth of a graph, and so also the treewidth of $M^c$ is at most $7(1+\lambda)(\lambda+\delta)$. It is easy to see that attaching any number of arborescences to $M^c$ does not increase the treewidth. Thus we obtain Theorem 4, which is the basis for our algorithm to solve $H$-DSN in case every pattern of $H$ is transitively equivalent to an almost-caterpillar.

In particular, when adding $\delta$ edges to the pattern of the DST problem, which is a single out-star, i.e., a 1-caterpillar, then the pattern becomes a member of $C_{1,\delta}$ and hence our result implies a linear treewidth bound of $O(\delta)$. The example given at the end of Section 2 also shows that there are patterns $H \in C_{1,\delta}$ for which every minimal solution has treewidth $\Omega(\lambda+\delta)$: just consider the case when $H$ is a cycle of length $\lambda+\delta$ (i.e., it contains a trivial caterpillar graph). One interesting question is whether the treewidth bound of $7(1+\lambda)(\lambda+\delta)$ in Theorem 4 is tight. We conjecture that the treewidth of any minimal solution to a pattern graph $H \in C_{1,\delta}$ actually is $O(\lambda+\delta)$.

**Proof (of Theorem 12).** Let $M$ be a minimal solution to a pattern $H \in C_{1,\delta}$. Since every pattern in $C_{1,\delta}$ has a transitively equivalent pattern in $C_{3,\delta}$ and replacing a pattern with a transitively equivalent pattern does not change the space of feasible solutions, we may assume that $H$ is actually in $C_{1,\delta}$, i.e., $H$ consists of a caterpillar of length at most $\lambda$ and $\delta$ additional edges.

The statement is trivial if $|E(H)| \leq \delta$. Otherwise, according to Definition 1, $H$ contains a $\lambda_0$-caterpillar for some $1 \leq \lambda_0 \leq \lambda$ and at most $\delta$ additional edges. Hence let us fix a set $F$ of at most $\delta$ edges of $H$, such that the remaining edges of $H$ form a $\lambda_0$-caterpillar $C$, for some $1 \leq \lambda_0 \leq \lambda$, with a path $(v_1, \ldots, v_{\lambda_0})$ on the roots of the stars $S_i$. We only consider the case when $C$ is an out-caterpillar as the other case is symmetric, i.e., every $S_i$ is an out-star. Define $I = H \setminus \bigcup_{i=1}^{\lambda_0} S_i$ to be all of $H$ except the stars. Note that $|E(I)| \leq \lambda + \delta$. We fix a subgraph $M_I$ of $M$ that is a minimal solution to the sub-pattern $I$, and for every $st \in E(I)$ we fix a path $P_{st}$ in $M_I$. Note that $M_I$ is the union of these at most $\lambda + \delta$ paths, since $M_I$ is a minimal solution. For each star $S_i$, let us consider a minimal solution $M_{S_i} \subseteq M$ to $S_i$; note that $M_{S_i}$ has to be an out-arborescence.

For $i \in \{1, \ldots, \lambda_0\}$, let $\ell_i$ be a leaf of $S_i$, and let $e_i$ be an edge of $M$. If $M \setminus e_i$ has no path from $v_i$ to $\ell_i$, then we say that $e_i$ is $\ell_i$-necessary. More generally, we say that $e_i$ is $i$-necessary if $e_i$ is $\ell_i$-necessary for some leaf $\ell_i$ of $S_i$. 
▶ Claim 13. Let $P$ be a path in $M$, and for some $i \in \{1, \ldots, \lambda_0\}$, let $W_i \subseteq E(M)$ contain all $i$-necessary edges $f$ for which $f \notin E(P)$, but the head of $f$ is a vertex of $P$. Then there exists one leaf $\ell$ of $S_i$ such that every $f \in W_i$ is $\ell$-necessary.

Using this observation, we identify the core $M^c$ of $M$ using the at most $\lambda + \delta$ paths $P_{st}$ that make up $M_I$, and then selecting an additional at most $\lambda_0$ paths for each $P_{st}$. To construct $M^c$ together with its pattern graph $H^c$, we initially let $M^c = M_I$ and $H^c = I$ and repeat the following step for every $st \in E(I)$ and $1 \leq i \leq \lambda_0$. For a given $st$ and $i$, let us check if there are $i$-necessary edges $f \notin E(P_{st})$ that have their heads on the path $P_{st} \subseteq M_I$. If so, then by Claim 13 all these edges are $\ell$-necessary for some leaf $\ell$ of $S_i$. We add an arbitrary path of $M$ from $v_i$ to $\ell$ (which contains all these edges) to $M^c$ and add the edge $v_i\ell$ to $H^c$. After repeating this step for every $st \in E(H)$ and $i$, we remove superfluous edges from $M^c$: as long as there is an edge $e \in E(M^c)$, which can be removed while maintaining feasibility for the pattern $H^c$, i.e., for every $vw \in E(H^c)$ there is a $v \to w$ path in $M^c$ not containing $e$, we remove $e$. Finally, we remove any isolated vertices from $M^c$.

Note that the resulting network $M^c$ is a minimal solution to $H^c$ by construction. Also note that $H^c$ contains at most $\lambda + \delta$ edges from $I$ and at most $\lambda_0 \leq \lambda$ additional edges for each edge of $I$, so that $|E(H^c)| \leq (1 + \lambda)(\lambda + \delta)$. We prove that the remaining graph $M^c \setminus E(M)$ consists of arborescences, each of which intersects $M^c$ only at the root. For this, we rely on the following key observation.

▶ Claim 14. If a vertex $u$ has at least two incoming edges in $M$, then every such edge is in the core $M^c$.

Proof. First we show that there is an $st \in E(I)$ such that every $s \to t$ path in $M$ goes through $u$. Suppose for contradiction that for every $st \in E(I)$ there is a path from $s$ to $t$ in $M$ avoiding $u$. Since $M$ is a minimal solution, the edges entering $u$ must then be needed for some stars $S_i$ of the pattern $H$ instead. Let $e$ and $f$ be two edges entering $u$. As $e$ and $f$ have the same head, they cannot be part of the same out-arborescence $M_{S_i}$. Therefore, there are indices $i < j$ such that (w.l.o.g.) $e$ is $i$-necessary and $f$ is $j$-necessary.

There is a path in $M$ from the root $v_i$ of $S_i$ to the root $v_j$ of $S_j$, due to the path $(v_1, \ldots, v_{\lambda_0})$ in the caterpillar $C \subseteq H$. Since path $(v_1, \ldots, v_{\lambda_0})$ is part of $I$, our assumption on $e$ and $f$ implies that there is a path $P$ in $M$ from $v_i$ to $v_j$ that avoids both $e$ and $f$. As $f \in E(M_{S_j})$, there is a path $Q$ in $M$ starting in $v_i$ and passing through $f$. This path cannot contain $e$, as $e$ and $f$ have the same head $u$. The existence of $P$ and $Q$ implies that $u$ can be reached from $v_i$ by a path through $v_j$ and $f$, avoiding the edge $e$. Thus for any edge $v_i\ell \in E(S_i)$, if there is a $v_i \to \ell$ path going through $e$ (and hence vertex $u$), then it can be rerouted to avoid $e$ and use edge $f$ instead. This however contradicts the fact that $e$ is $i$-necessary.

We have proved that there is an $st \in E(I)$ such that every $s \to t$ path in $M$ goes through $u$. Suppose that there is an edge $e \notin E(M^c)$ entering $u$. If $e$ is needed for some $s't' \in E(I)$ in $M$, then $e$ is also present in $M^c$, and we are done. Otherwise, as $M$ is a minimal solution, edge $e$ is $i$-necessary for some $i \in \{1, \ldots, \lambda_0\}$. Consider now the step in the construction of $M^c$ when we considered $st \in E(I)$ and integer $i$. As we have shown, the $s \to t$ path $P_{st}$ goes through $u$. Thus $e$ is an $i$-necessary edge not in $E(P_{st})$ such that its head is on $P_{st}$. This means that we identified a leaf $\ell$ of $S_i$ such that $e$ is $\ell$-necessary, introduced $v_i\ell$ into $H^c$, and added a $v_i \to \ell$ path to $H^c$, which had to contain $e$. Moreover, since all paths from $v_i$ to $\ell$ in $M$ pass through $e$, edge $e$ then remains in $M^c$ when removing superfluous edges. ◀
We are now ready to show that every component of the remaining part is an out-arborescence and intersects the core only in a single vertex.

▶ Claim 15. The remaining graph $M^+ := M \setminus E(M^c)$ is a forest of out-arborescences, each of which intersects $M^c$ only at the root.

Since we have already established that $M^c$ is a minimal solution to $H^c$ with $|E(H^c)| \leq (1 + \lambda)(\lambda + \delta)$, Claim 15 completes the proof of Theorem 12. ◀

4 Characterizing the hard cases

We now turn to proving the second part of Theorem 2, i.e., that $H$-DSN is $W[1]$-hard for every class $H$ where the patterns are not transitively equivalent to almost-caterpillars.

▶ Theorem 16. Let $H$ be a recursively enumerable class of patterns for which there are no constants $\lambda$ and $\delta$ such that $H \subseteq C^{*,\delta}_\lambda$. Then the problem $H$-DSN is $W[1]$-hard for parameter $k$.

A major technical simplification is to assume that the class $H$ is closed under identifying terminals and transitive equivalence. As we show in Section 4.1, this assumption is not really restrictive: it is sufficient to prove hardness for the closure of $H$ under identification and transitive equivalence, since any $W[1]$-hardness result for the closure can be transferred to $H$.

For classes closed under these operations, it is possible to give an elegant characterization of the classes that are not almost-caterpillars. There are only a few very specific reasons why a class $H$ is not in $C^{*,\delta}_\lambda$ for any $\lambda$ and $\delta$: either $H$ contains every directed cycle, or $H$ contains every “pure diamond,” or $H$ contains every “flawed diamond” (see Section 4.2 for the precise definitions). Then in Section 4.3, we provide a $W[1]$-hardness proof for each of these cases, completing the hardness part of Theorem 2.

4.1 Closed classes

We define the operation of identifying terminals in the following way: given a partition $V$ of the vertex set $V(H)$ of a pattern graph $H$, each set $W \in V$ is identified with a single vertex of $W$, after which any resulting isolated vertices and self-loops are removed, while parallel edges having the same head and tail are replaced by only one copy of that edge. A class of patterns is closed under this operation if for any pattern $H$ in the class, all patterns that can be obtained by identifying terminals are also in the class. Similarly, we say that a class $H$ is closed under transitive equivalence if whenever $H$ and $H'$ are two transitively equivalent patterns such that $H \in H$, then $H'$ is also in $H$. The closure of the class $H$ under identifying terminals and transitive equivalence is the smallest closed class $H' \supseteq H$. It is not difficult to see that any member of the closure can be obtained by a replacement with a transitively equivalent pattern and a single application of identifying terminals.

The following lemma shows that if we want to prove $W[1]$-hardness for a class, then it is sufficient to prove hardness for its closure. More precisely, due to an slight technicality, the actual statement we prove is that it is sufficient to prove $W[1]$-hardness for a decidable subclass of the closure.

▶ Lemma 17. Let $H$ be a recursively enumerable class of patterns, let $H'$ be the closure of $H$ under identifying terminals and transitive equivalence, and let $H''$ be a decidable subclass of $H'$. There is a parameterized reduction from $H''$-DSN to $H$-DSN with parameter $k$. 
Figure 2 The obstruction appearing in Lemma 19: a) a directed cycle of length 4, b) a pure 4-out-diamond, c) a flawed 4-out-diamond, d) a pure 4-in-diamond, e) a flawed 4-in-diamond.

4.2 Obstructions: SCCs and diamonds

To show the hardness for a closed class that is not the subset of $C_{\lambda,\delta}^*$ for any $\lambda$ and $\delta$, we will characterize such a class in terms of the occurrence of arbitrarily large cycles, and another class of patterns called “diamonds” (cf. Figure 2).

Definition 18. A pure $\alpha$-diamond graph is constructed as follows. Take a vertex set $L$ of size $\alpha \geq 1$, and two additional vertices $r_1$ and $r_2$. Now add edges such that $L$ is the leaf set of either two in-stars or two out-stars $S_1$ and $S_2$ with roots $r_1$ and $r_2$, respectively. If we add an additional vertex $x$ with edges $r_1x$ and $r_2x$ if $S_1$ and $S_2$ are in-stars, and edges $xr_1$ and $xr_2$ otherwise, the resulting graph is a flawed $\alpha$-diamond. We refer to both pure $\alpha$-diamonds and flawed $\alpha$-diamonds as $\alpha$-diamonds. If $S_1$ and $S_2$ are in-stars we also refer to the resulting $\alpha$-diamonds as in-diamonds, and otherwise as out-diamonds.

The goal of this section is to prove the following useful characterization precisely describing classes that are not almost-caterpillars.

Lemma 19. Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive closure. Exactly one of the following statements is true:

- $\mathcal{H} \subseteq C_{\lambda,\delta}^*$ for some constants $\lambda$ and $\delta$.
- $\mathcal{H}$ contains every directed cycle, or every pure in-diamond, or every pure out-diamond, or every flawed in-diamond, or every flawed out-diamond.

For the proof of Theorem 16, we only need the fact that at least one of these two statements hold: if the class $\mathcal{H}$ is not in $C_{\lambda,\delta}^*$, then we can prove hardness by observing that $\mathcal{H}$ contains one of the hard classes. For the sake of completeness, we give a simple proof that the two statements cannot hold simultaneously in the full version of this extended abstract.

Showing that at least one of the two statements of Lemma 19 hold is not as easy to prove. First, the following two lemmas show how a large cycle or a large diamond can be identified if certain structures appear in a pattern. The main part of the proof is to show that if $\mathcal{H}$ contains patterns that are arbitrarily far from being a caterpillar, then one of these two lemmas can be invoked (see Lemma 22).

Lemma 20. Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive closure. If some $H \in \mathcal{H}$ contains a matching of size $\alpha$, then $\mathcal{H}$ contains a directed cycle of length $\alpha$.

Proof. A matching of a graph is a subset $M$ of its edges such that no two edges of $M$ share a vertex. A matching $e_1, \ldots, e_\alpha$ of $\alpha$ edges can be transformed into a cycle of length $\alpha$ by identifying the head of $e_1$ and tail of $e_{\alpha+1}$ (and the head of $e_\alpha$ with the tail of $e_1$). All remaining vertices that do not belong to the cycle can then be identified with any vertex of the cycle, so that the resulting graph consists of the cycle and some additional edges. Since $\mathcal{H}$ is closed under identifying terminals, this graph would be contained in $\mathcal{H}$. As this graph is strongly connected and $\mathcal{H}$ is closed also under transitive equivalence, we can conclude that $\mathcal{H}$ contains a cycle of length $\alpha$. ◀
Next we give a sufficient condition for the existence of large diamonds. We say that an edge $uv$ of a graph $H$ is transitivity non-redundant if there is no $u \rightarrow v$ path in $H \setminus uv$.

**Lemma 21.** Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive equivalence. Let $H \in \mathcal{H}$ be a pattern graph that contains two out-stars (or two in-stars) $S_1$ and $S_2$ as induced subgraphs, with at least $\alpha$ edges each and roots $r_1$ and $r_2$, respectively. The class $\mathcal{H}$ contains an $\alpha$-diamond if

1. $H$ contains neither a path from $r_1$ to $r_2$, nor from $r_2$ to $r_1$,
2. the leaves of $S_1$ and $S_2$ have out-degree 0 (if $S_1$ and $S_2$ are out-stars) or in-degree 0 (if $S_1$ and $S_2$ are in-stars), and
3. the edges of the stars are transitivity non-redundant.

To show that at least one of the two statements of Lemma 19 hold, we prove that if the second statement is false, then the first statement is true. That is, if $\mathcal{H}$ does not contain all cycles (i.e., there is an $\alpha_1$ such that $\mathcal{H}$ contains no cycle larger than $\alpha_1$), $\mathcal{H}$ does not contain all pure out-diamonds (i.e., there is an $\alpha_2$ such that $\mathcal{H}$ contains no pure out-diamond larger than $\alpha_2$), etc., then $\mathcal{H} \subseteq C_{\alpha, \delta}$ for some constants $\lambda$ and $\delta$. In other words, if we let $\alpha$ be the maximum of $\alpha_1$, $\alpha_2$, etc., then we may assume that $\mathcal{H}$ contains no pure of flawed $\alpha$-diamond or cycle of length $\alpha$, and we need to prove $\mathcal{H} \subseteq C_{\alpha, \delta}$ under this assumption. Thus the following lemma completes the proof of Lemma 19.

**Lemma 22.** Let $\mathcal{H}$ be a class of pattern graphs that is closed under identifying terminals and transitive equivalence. If for some integer $\alpha$ the class $\mathcal{H}$ contains neither a pure $\alpha$-diamond, flawed $\alpha$-diamond, nor a cycle of length $\alpha$, then there exist constants $\lambda$ and $\delta$ (depending on $\alpha$) such that $\mathcal{H} \subseteq C_{\lambda, \delta}$.

**Proof.** Suppose that there is such an integer $\alpha$. Let $\lambda := 2\alpha$ and $\delta := 4\alpha^3 + 6\alpha^2$. Given any $H' \in \mathcal{H}$, we show how a transitively equivalent pattern $H \in C_{\lambda, \delta}$ can be constructed, implying that $H'$ belongs to $C_{\lambda, \delta}$. A vertex cover of a graph is a subset $X$ of its vertices such that every edge is incident to a vertex of $X$. By Lemma 20, $H'$ cannot contain a matching of size $\alpha$. It is well-known that if a graph has no matching of size $\alpha$, then it has a vertex cover of size at most $2\alpha$ (take the endpoints of any maximal matching). Let us fix a vertex cover $X$ of $H'$ having size at most $2\alpha$.

To obtain $H$ from $H'$, we start with a graph $H$ on $V(H')$ having no edges and perform the following three steps.

1. Let us take the transitive closure on the vertex set $X$ in $H'$, i.e., let us introduce into $H$ every edge $uv$ with $u, v \in X$ such that there is a $u \rightarrow v$ path in $H'$.
2. Let us add all edges $uv$ of $H'$ to $H$ for which $u \notin X$ or $v \notin X$.
3. Fixing an ordering of the edges introduced in step 2, we remove transitively redundant edges: following this order, we subsequently remove those edges $uv$ for which there is a path from $u$ to $v$ in the remaining graph $H$ that is not the edge $uv$ itself.

It is clear that $H$ is transitively equivalent to $H'$. Note that $X$ is a vertex cover of $H$ as well, and hence its complement $I = V(H) \setminus X$ is an independent set, i.e., no two vertices of $I$ are adjacent. Let $E_I \subseteq E(H)$ be the set of edges between $X$ and $I$. In the rest of the proof, we argue that the resulting pattern $H$ belongs to $C_{\lambda, \delta}$. We show that $H$ can be decomposed into a path $P = (v_1, \ldots, v_\lambda)$ in $X$, a star $S_{v_i}$ centered at each $v_i$ using the edges in $E_I$, and a small set of additional edges. This small set of additional edges is constructed in three steps, by considering a sequence of larger and larger sets $F_1 \subseteq F_2 \subseteq F_3$.

As $E_I$ consists of edges between $X$ and $I$, it can be partitioned into a set of stars with roots in $X$. The following claim shows that almost all of these edges are directed towards $X$ or almost all of them are directed away from $X$. 


Claim 23. Either there are less than $2\alpha^2$ edges $uv$ in $E_I$ with head in $X$, or less than $2\alpha^2$ edges $vw$ in $E_I$ with tail in $X$.

Assume that the former case of Claim 23 is true, so that the number of edges in $E_I$ with heads in $X$ is bounded by $2\alpha^2$; the other case can be handled symmetrically. We will use the out-stars spanned by $E_I$ for the caterpillar, which means that we obtain an out-caterpillar. We use the set $F_1$ to account for the edges in $E_I$ with heads in $X$. Additionally, we will also introduce into $F_1$ those edges in $E_I$ with tails in $X$ that are adjacent to an edge of the former type. Formally, for any edge $uv \in E_I$ with $v \in X$, we introduce into $F_1$ every edge of $E_I$ incident to $u$. After this step, $F_1$ contains less than $4\alpha^3$ edges, since there are less than $2\alpha^2$ edges $uv \in E_I$ with $v \in X$ and $u$ can only be adjacent to vertices in $X$, which has size less than $2\alpha$.

For any vertex $v \in X$, let $S_v$ denote the out-star formed by the edges of $E_I \setminus F$ incident to $v$. Let $X' \subseteq X$ contain those vertices $v \in X$ for which $S_v$ has at least $\alpha$ leaves.

Claim 24. For any two distinct $u, v \in X'$, at least one of $uv$ and $vu$ is in $H$, and the stars $S_u$ and $S_v$ are vertex disjoint.

We extend $F_1$ to $F_2$ by adding all edges of stars $S_v$ with $v \in X \setminus X'$ to $F_2$. Since $X$ contains less than $2\alpha$ vertices and we extend $F_1$ only by stars with less than $\alpha$ edges, this step adds less than $2\alpha^2$ edges, i.e., $|F_2| \leq |F_1| + 2\alpha^2 = 4\alpha^3 + 2\alpha^2$.

By Claim 24, $X'$ induces a semi-complete directed graph in $H$, i.e., at least one of the edges $uv$ and $vu$ exists for every pair $u, v \in X'$. It is well-known that every semi-complete directed graph contains a Hamiltonian path (e.g., [16, Chapter 10, Exercise 1]), and so there is a path $P = (v_1, \ldots, v_{\lambda_0})$ with $\lambda_0 = |X'| \leq 2\alpha = \lambda$ in $H$ on the vertices of $X'$. We extend $F_2$ to $F_3$ by including any edge induced by vertices of $X'$ that is not part of $P$. There are less than $4\alpha^2$ such edges, and hence we have $|F_3| \leq |F_2| + 4\alpha^2 \leq 4\alpha^3 + 6\alpha^2 = \delta$. The edges of $H$ not in $F_3$ span the path $P$ and disjoint out-stars $S_{v_i}$ with $i \in \{1, \ldots, \lambda_0\}$, i.e., they form a $\lambda_0$-caterpillar. This proves that $H \in \mathcal{C}_{\lambda, \delta}$ and hence $H' \in \mathcal{C}^*_{\lambda, \delta}$, what we had to show.

4.3 Reductions

Lemma 19 implies that in order to prove Theorem 16, we need $W[1]$-hardness proofs for the class of all directed cycles, the class of all pure in-diamonds, the class of all pure out-diamonds, etc. We provide these hardness proofs and then formally show that they imply Theorem 16.

Let us first consider the case when $\mathcal{H}$ is the class of all directed cycles. Recall that, given an arc-weighted directed graph $G$ and a set $R \subseteq V(G)$ of terminals, the STRONGLY CONNECTED STEINER SUBGRAPH (SCSS) problem asks for a minimum-cost subgraph that is strongly connected and contains every terminal in $R$. This problem is known to be $W[1]$-hard parameterized by the number $k := |R|$ of terminals [21]. We can reduce SCSS to an instance of DSN where the pattern $H$ is a directed cycle on $R$, which expresses the requirement that all the terminals are in the same strongly connected component of the solution. Thus the $W[1]$-hardness of SCSS immediately implies the $W[1]$-hardness of $\mathcal{H}$-DSN if $\mathcal{H}$ contains all directed cycles.

Claim 25 (follows from [21]). If $\mathcal{H}$ is the class of directed cycles, then $\mathcal{H}$-DSN is $W[1]$-hard parameterized by the number of terminals.

Next we turn our attention to classes containing all diamonds. The following reductions are from the $W[1]$-hard MULTICOLOURED CLIQUE problem [19], in which an undirected graph together with a partition $\{V_1, \ldots, V_k\}$ of its vertices into $k$ sets is given, such that for
any \( i \in \{1, \ldots, k\} \) no two vertices of \( V_i \) are adjacent. The aim is to find a clique of size \( k \), i.e. a set of pairwise adjacent vertices \( \{w_1, \ldots, w_k\} \) with \( w_i \in V_i \) for each \( i \in \{1, \ldots, k\} \).

▶ **Lemma 26.** If \( \mathcal{H} \) is the class of all pure out-diamonds, then \( \mathcal{H} \)-DSN is \( W[1] \)-hard parameterized by the number of terminals. The same holds if \( \mathcal{H} \) is the class of all pure in-diamonds.

The reduction for the case when the pattern is a flawed \( \alpha \)-diamond is essentially the same as the one for pure \( \alpha \)-diamonds, as we show next.

▶ **Lemma 27.** If \( \mathcal{H} \) is the class of all flawed out-diamonds, then \( \mathcal{H} \)-DSN is \( W[1] \)-hard parameterized by the number of terminals. The same holds if \( \mathcal{H} \) is the class of all flawed in-diamonds.

Given the three reductions above, we can now prove Theorem 16, based on the additional reduction given in Lemma 17. We defer the final proof to the full version of this extended abstract.

---

**References**


Characterization of Tractability for Directed Steiner Network


The Parameterized Hardness of the $k$-Center Problem in Transportation Networks

Andreas Emil Feldmann\textsuperscript{1}\textsuperscript{,\textcopyright} · Dániel Marx\textsuperscript{2}

Received: 22 November 2018 / Accepted: 24 January 2020 / Published online: 3 February 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

In this paper we study the hardness of the $k$-CENTER problem on inputs that model transportation networks. For the problem, a graph $G = (V, E)$ with edge lengths and an integer $k$ are given and a center set $C \subseteq V$ needs to be chosen such that $|C| \leq k$. The aim is to minimize the maximum distance of any vertex in the graph to the closest center. This problem arises in many applications of logistics, and thus it is natural to consider inputs that model transportation networks. Such inputs are often assumed to be planar graphs, low doubling metrics, or bounded highway dimension graphs. For each of these models, parameterized approximation algorithms have been shown to exist. We complement these results by proving that the $k$-CENTER problem is W[1]-hard on planar graphs of constant doubling dimension, where the parameter is the combination of the number of centers $k$, the highway dimension $h$, and the pathwidth $p$. Moreover, under the exponential time hypothesis there is no $f(k, p, h) \cdot n^{o(p + \sqrt{k + h})}$ time algorithm for any computable function $f$. Thus it is unlikely that the optimum solution to $k$-CENTER can be found efficiently, even when assuming that the input graph abides to all of the above models for transportation networks at once! Additionally we give a simple parameterized $(1 + \epsilon)$-approximation algorithm for inputs of doubling dimension $d$ with runtime $(k^k / \epsilon^{O(kd)}) \cdot n^{O(1)}$. This generalizes a previous result, which considered inputs in $D$-dimensional $L_q$ metrics.

Keywords $k$-Center problem · Parameterized complexity · Planar graphs · Highway dimension · Doubling dimension · Pathwidth
1 Introduction

Given a graph $G = (V, E)$ with positive edge lengths $\ell : E \rightarrow \mathbb{Q}^+$, the $k$-Center problem asks to find $k$ center vertices such that every vertex of the graph is as close as possible to one of the centers. More formally, a solution to $k$-Center is a set $C \subseteq V$ of centers such that $|C| \leq k$. If $\text{dist}(u, v)$ denotes the length of the shortest path between $u$ and $v$ according to the edge lengths $\ell$, the objective is to minimize the cost $\rho = \max_{u \in V} \min_{v \in C} \text{dist}(u, v)$ of the solution $C$. While this is the standard way of defining the problem, throughout this paper we will rather think of it as covering the graph with balls of minimum radius. That is, let $B_v(r) = \{u \in V \mid \text{dist}(u, v) \leq r\}$ be the ball of radius $r$ around $v$. The cost of a solution $C$ equivalently is the smallest value $\rho$ for which $\bigcup_{v \in C} B_v(\rho) = V$. The $k$-Center problem has numerous applications in logistics where easily accessible locations need to be chosen in a network under a budget constraint. For instance, a budget may be available to build $k$ hospitals, shopping malls, or warehouses. These should be placed so that the distance from each point on the map to the closest facility is minimized.

The $k$-Center problem is NP-hard [29], and so approximation algorithms [29, 30] as well as parameterized algorithms [9, 12] have been developed for this problem. The former are algorithms that use polynomial time to compute an $\alpha$-approximation, i.e., a solution that is at most $\alpha$ times worse than the optimum. For the latter, a parameter $q$ is given as part of the input, and an optimum solution is computed in $f(q) \cdot n^{O(1)}$ time for some computable function $f$ independent of the input size $n$. The rationale behind such an algorithm is that it solves the problem efficiently in applications where the parameter is small. If such an algorithm exists, the corresponding problem is called fixed-parameter tractable (FPT) for $q$. Another option is to consider parameterized approximation algorithms [23, 25], which compute an $\alpha$-approximation in $f(q) \cdot n^{O(1)}$ time for some parameter $q$.

By a result of Hochbaum and Shmoys [20], on general input graphs, a polynomial time 2-approximation algorithm exists, and this approximation factor is also best possible, unless $P = \text{NP}$. A natural parameter for $k$-Center is the number of centers $k$, for which however the problem is W[2]-hard [10], and is thus unlikely to be FPT. In fact it is even W[2]-hard [15] to compute a $(2 - \varepsilon)$-approximation for any $\varepsilon > 0$, and thus parametrizing by $k$ does not help to overcome the polynomial-time inapproximability. For structural parameters such as the vertex-cover number or the feedback-vertex-set number the problem remains W[1]-hard [22], even when combining with the parameter $k$. For each of the two more general structural parameters treewidth and cliquewidth, an efficient parameterized approximation scheme (EPAS) was shown to exist [22], i.e., a $(1 + \varepsilon)$-approximation can be computed in $f(\varepsilon, w) \cdot n^{O(1)}$ time for any $\varepsilon > 0$, if $w$ is either the treewidth or the cliquewidth, and $n$ is the number of vertices.

Arguably however, graphs with low treewidth or cliquewidth do not model transportation networks well, since grid-like structures with large treewidth and cliquewidth can occur in road maps of big cities. As we focus on applications for $k$-Center in logistics, here we consider more natural models for transportation
networks. These include planar graphs, low doubling metrics such as the Euclidean or Manhattan plane, or the more recently studied low highway dimension graphs. Our main result is that \(k\)-Center is \(W[1]\)-hard on all of these graph classes combined, even if adding \(k\) and the pathwidth as parameters (note that the pathwidth is a stronger parameter than the treewidth). Before introducing these graph classes, let us formally state our theorem.

**Theorem 1** Even on planar graphs with edge lengths of doubling dimension \(O(1)\), the \(k\)-Center problem is \(W[1]\)-hard for the combined parameter \((k, p, h)\), where \(p\) is the pathwidth and \(h\) the highway dimension of the input graph. Moreover, under ETH there is no \(f(k, p, h) \cdot n^{o(p + \sqrt{k + h})}\) time algorithm\(^1\) for the same restriction on the input graphs, for any computable function \(f\).

A planar graph can be drawn in the plane without crossing edges. Such graphs constitute a realistic model for road networks, since overpasses and tunnels are relatively rare. It is known [28] that also for planar graphs no \((2 - \epsilon)\)-approximation can be computed in polynomial time, unless \(P = NP\). On the positive side, \(k\)-Center is FPT [10] on unweighted planar graphs for the combined parameter \(k\) and the optimum solution cost \(\rho\). However, typically if \(k\) is small then \(\rho\) is large and vice versa, and thus the applications for this combined parameter are rather limited. If the parameter is only \(k\), then an \(n^{O(\sqrt{k})}\) time algorithm exists for planar graphs [26]. By a very recent result [18] the \(k\)-Center problem on planar graphs with positive edge lengths admits an efficient polynomial-time bicriteria approximation scheme, which for any \(\epsilon > 0\) in \(f(\epsilon) \cdot n^{O(1)}\) time computes a solution that uses at most \((1 + \epsilon)k\) centers and approximates the optimum with at most \(k\) centers within a factor of \(1 + \epsilon\). This algorithm implies an EPAS for parameter \(k\) on planar graphs with edge lengths, since setting \(\epsilon = \min\{\epsilon', \frac{1}{2k}\}\) forces the algorithm to compute a \((1 + \epsilon')\)-approximation in \(f(k, \epsilon') \cdot n^{O(1)}\) time using at most \((1 + \epsilon)k \leq k + \frac{1}{2}\) centers, i.e., at most \(k\) centers as \(k\) is an integer. This observation is complemented by our hardness result showing that it is necessary to approximate the solution when parametrizing by \(k\) in planar graphs with edge lengths.

**Definition 2** The doubling dimension of a metric \((X, \text{dist})\) is the smallest \(d \in \mathbb{R}\) such that for any \(r > 0\), every ball of radius \(2r\) is contained in the union of at most \(2^d\) balls of radius \(r\). The doubling dimension of a graph is the doubling dimension of its shortest-path metric.

Since a transportation network is embedded on a large sphere (namely the Earth), a reasonable model is to assume that the shortest-path metric abides to the Euclidean \(L_2\)-norm. In cities, where blocks of buildings form a grid of streets, it is reasonable to assume that the distances are given by the Manhattan \(L_1\)-norm. Every metric for which the distance function is given by the \(L_q\)-norm in \(D\)-dimensional space \(\mathbb{R}^D\) has doubling

---

\(^1\) Here \(o(p + \sqrt{k + h})\) means \(g(p + \sqrt{k + h})\) for any function \(g\) such that \(g(x) \in o(x)\).
dimension $O(D)$. Thus a road network, which is embedded into $\mathbb{R}^2$ can reasonably be assumed to have constant doubling dimension. It is known [24] that $k$-CENTER is $\text{W}[1]$-hard for parameter $k$ in two-dimensional Manhattan metrics. Also, no polynomial time $(2 - \varepsilon)$-approximation algorithm exists for $k$-CENTER in two-dimensional Manhattan metrics [14], and no $(1.822 - \varepsilon)$-approximation for two-dimensional Euclidean metrics [14]. On the positive side, Agarwal and Procopiuc [4] showed that for any $L_q$ metric in $D$ dimensions, the $k$-CENTER problem can be solved optimally in $n^{O(k^{1-1/D})}$ time, and an EPAS exists for the combined parameter $(\varepsilon, k, D)$. We generalize the latter to any metric of doubling dimension $d$, as formalized by the following theorem.

**Theorem 3** Given a metric of doubling dimension $d$ and $\varepsilon > 0$, a $(1 + \varepsilon)$-approximation for $k$-CENTER can be computed in $(k^2 / \varepsilon^{O(kd)}) \cdot n^{O(1)}$ time.

Theorem 1 complements this result by showing that it is necessary to approximate the cost of the solution if parametrizing by $k$ and $d$.

**Definition 4** The highway dimension of a graph $G$ is the smallest $h \in \mathbb{N}$ such that, for some universal constant $c \geq 4$, for every $r \in \mathbb{R}^+$ and every ball $B_{cr}(v)$ of radius $cr$, there is a set $H \subseteq B_{cr}(v)$ of hubs such that $|H| \leq h$ and every shortest path of length more than $r$ lying in $B_{cr}(v)$ contains a hub of $H$.

The highway dimension was introduced by Abraham et al. [1] as a formalization of the empirical observation by Bast et al. [5, 6] that in a road network, starting from any point $A$ and travelling to a sufficiently far point $B$ along the quickest route, one is bound to pass through some member of a sparse set of “access points”, i.e., the hubs. In contrast to planar and low doubling graphs, the highway dimension has the potential to model not only road networks but also more general transportation networks such as those given by air-traffic or public transportation. This is because in such networks longer connections tend to be serviced through larger and sparser stations, which act as hubs. Abraham et al. [1] were able to prove that certain shortest-path heuristics are provably faster in low highway dimension graphs than in general graphs. They specifically chose the constant $c = 4$ in their original definition, but later work by Feldmann et al. [17] showed that when choosing any constant $c > 4$ in the definition, the structure of the resulting graphs can be exploited to obtain quasi-polynomial time approximation schemes for problems such as TRAVELING SALESMAN or FACILITY LOCATION. Note that increasing the constant $c$ in Definition 4 restricts the class of graphs further. Moreover, as shown by Feldmann et al. [17, Section 9], the highway dimension of a graph according to Definition 4 can grow arbitrarily large by just a small change in the constant $c$: for any $c$ there is a graph of highway dimension 1 when using $c$ in Definition 4, which however has highway dimension $\Omega(n)$ for any constant larger than $c$.

We remark that these graphs have unbounded doubling dimension, and that an upper bound of $O(hc^d)$ on the highway dimension of any graph using constant $c$ in Definition 4 can be shown, if the doubling dimension is $d$ and $h$ is the highway dimension using constant 4.
highway dimension exist as well [1–3] (see Feldmann et al. [17, Section 9] and Blum [8] for detailed discussions).

Later, Becker et al. [7] used the framework introduced by Feldmann et al. [17] to show that whenever \( c > 4 \) there is an EPAS for \( k\text{-CENTER} \) parameterized by \( \epsilon, k, \) and \( h \). Note that the highway dimension is always upper bounded by the vertex-cover number, as every edge of any non-trivial path is incident to a vertex cover. Hence the aforementioned \( \mathsf{W}[1] \)-hardness result by Katsikarelis et al. [22] for the combined parameter \( k \) and the vertex-cover number proves that it is necessary to approximate the optimum when using \( k \) and \( h \) as the combined parameter. When parametrizing only by the highway dimension but not \( k \), it is not even known if a parameterized approximation scheme (PAS) exists, i.e., an \( f(\epsilon, h) \cdot n^{g(\epsilon)} \) time \((1 + \epsilon)\)-approximation algorithm for some computable functions \( f, g \). However, under the exponential time hypothesis (ETH) [9], by [15] there is no algorithm with doubly exponential \( 2^{2^{\Theta(h)}} \cdot n^{O(1)} \) runtime computing a \((2 - \epsilon)\)-approximation for any \( \epsilon > 0 \). The same paper [15] also presents a \( 3/2 \)-approximation for \( k\text{-CENTER} \) with runtime \( 2^{O(kh \log h)} \cdot n^{O(1)} \) for a more general definition of the highway dimension than the one given in Definition 4 (based on so-called shortest path covers). In contrast to the result of Becker et al. [7], it is not known whether a PAS exists when combining this more general definition of \( h \) with \( k \) as a parameter. Theorem 1 complements these results by showing that even on planar graphs of constant doubling dimension, for the combined parameter \((k, h)\) no fixed-parameter algorithm exists, unless \( \mathsf{FPT} = \mathsf{W}[1] \). Therefore approximating the optimum is necessary, regardless of whether \( h \) is according to Definition 4 or the more general one from [15], and regardless of how restrictive Definition 4 is made by increasing the constant \( c \).

**Definition 5** A path decomposition of a graph \( G = (V, E) \) is a path \( P \) each of whose nodes \( v \) is labelled by a bag \( K_v \subseteq V \) of vertices of \( G \), and has the following properties:

(a) \( \bigcup_{v \in V(P)} K_v = V \),
(b) for every edge \( \{u, w\} \in E \) there is a node \( v \in V(P) \) such that \( K_v \) contains both \( u \) and \( w \),
(c) for every \( v \in V \) the set \( \{u \in V(P) \mid v \in K_u\} \) induces a connected subpath of \( P \).

The width of the path decomposition is \( \max \{|K_v| - 1 \mid v \in V(P)\} \). The pathwidth \( p \) of a graph \( G \) is the minimum width among all path decompositions for \( G \).

The pathwidth of a graph is always at least as large as its treewidth (for which the path \( P \) in the above definition is replaced by a tree). Thus, as mentioned above, arguably, bounded pathwidth graphs are not a good model for transportation networks. Also it is already known that \( k\text{-CENTER} \) is \( \mathsf{W}[1] \)-hard for this parameter, even when combining it with \( k \) [22]. We include this well-studied parameter here nonetheless, since the reduction of our hardness result in Theorem 1 implies that \( k\text{-CENTER} \) is \( \mathsf{W}[1] \)-hard even for planar graphs with edge lengths when combining any of the parameters \( k, h, d, \) and \( p \). As noted by Feldmann et al. [17] and Blum
these parameters are not bounded in terms of each other, i.e., they are incomparable. Furthermore, the doubling dimension is in fact bounded by a constant in Theorem 1. Hence, even if one were to combine all the models presented above and assume that a transportation network is planar, is embeddable into some metric of constant doubling dimension, has bounded highway dimension, and even has bounded pathwidth, the \( k \)-CENTER problem cannot be solved efficiently, unless \( \text{FPT}=\text{W}[1] \). Thus it seems unavoidable to approximate the problem in transportation networks, when developing fast algorithms.

### 1.1 Related Work

The above mentioned efficient bicriteria approximation scheme [18] improves on a previous (non-efficient) bicriteria approximation scheme [13], which for any \( \epsilon > 0 \) and planar input graph with edge lengths computes a \((1+\epsilon)\)-approximation with at most \((1+\epsilon)k\) centers in time \( n^{f(\epsilon)} \) for some function \( f \) (note that in contrast to above, such an algorithm does not imply a PAS for parameter \( k \)). The paper by Demaine et al. [10] on the \( k \)-CENTER problem in unweighted planar graphs also considers the so-called class of map graphs, which is a superclass of planar graphs that is not minor-closed. They show that the problem is FPT on unweighted map graphs for the combined parameter \((k, \rho)\). Also for the tree-depth, \( k \)-CENTER is FPT [22]. Another parameter related to transportation networks is the skeleton dimension, for which it was recently shown [8] that, under ETH, no \( 2^{\sqrt{s}} n^{O(1)} \) time algorithm can compute a \((2-\epsilon)\)-approximation for any \( \epsilon > 0 \), if the skeleton dimension is \( s \). It is not known whether this parameter yields any approximation schemes when combined with for instance \( k \), as is the case for the highway dimension.

A closely related problem to \( k \)-CENTER is the \( \rho \)-DOMINATING SET problem, in which \( \rho \) is given and the number \( k \) of centers covering a given graph with \( k \) balls of radius \( \rho \) needs to be minimized. As this generalizes the DOMINATING SET problem, no \((\ln(n)-\epsilon)\)-approximation is possible in polynomial time [11], unless \( \text{P}=\text{NP} \), and computing an \( f(k) \)-approximation is \( \text{W}[1] \)-hard [21] when parametrizing by \( k \), for any computable function \( f \).

### 2 The Reduction

In this section we give a reduction from the GRID TILING WITH INEQUALITY (GT\( \leq \)) problem, which was introduced by Marx and Sidiropoulos [27] and is defined as follows. Given \( \kappa^2 \) non-empty sets \( S_{i,j} \subseteq [n]^2 \) of pairs of integers,\(^3\) where \( i,j \in [\kappa] \), the task is to select one pair \( s_{i,j} \in S_{i,j} \) for each set such that

- if \( s_{i,j} = (a, b) \) and \( s_{i+1,j} = (a', b') \) for \( i \leq \kappa - 1 \) then \( a \leq a' \), and
- if \( s_{i,j} = (a, b) \) and \( s_{i,j+1} = (a', b') \) for \( j \leq \kappa - 1 \) then \( b \leq b' \).

---

\(^3\) For any positive integer \( q \), throughout this article \([q]\) means \( \{1, \ldots, q\} \).
The GT_≤ problem is W[1]-hard [9] for parameter κ, and moreover, under ETH has no $f(\kappa) \cdot n^{o(\kappa)}$ time algorithm for any computable function $f$.

2.1 Construction

Given an instance $I$ of GT_≤ with $\kappa^2$ sets, we construct the following graph $G_I$. First, for each set $S_{ij},$ where $1 \leq i, j \leq \kappa,$ we fix an arbitrary order on its elements, so that $S_{ij} = \{s_1, \ldots, s_\sigma\},$ where $\sigma \leq n^2.$ We then construct a gadget $G_{ij}$ for $S_{ij},$ which contains a cycle $O_{ij}$ of length $16n^2 + 4$ in which each edge has length 1 (see Fig. 1a). Additionally we introduce five vertices $x^1_{ij}, x^2_{ij}, x^3_{ij}, x^4_{ij},$ and $y_{ij}.$ If $O_{ij} = (v_1, v_2, \ldots, v_{16n^2+4}, v_1)$ then we connect these five vertices to the cycle as follows. The vertex $y_{ij}$ is adjacent to the four vertices $v_1, v_{4n^2+2}, v_{8n^2+3},$ and $v_{12n^2+4},$ with edges of length $2n^2 + 1$ each. For every $\tau \in [\sigma]$ and $s_\tau \in S_{ij},$ if $s_\tau = (a, b)$ we add the four edges

- $x^1_{ij}v_\tau$ of length $\ell'_a = 2n^2 - \frac{a}{n+1},$
- $x^2_{ij}v_{\tau+4n^2+1}$ of length $\ell''_b = 2n^2 + \frac{b}{n+1} - 1,$
- $x^3_{ij}v_{\tau+8n^2+2}$ of length $\ell''_a = 2n^2 + \frac{a}{n+1} - 1,$ and
- $x^4_{ij}v_{\tau+12n^2+3}$ of length $\ell'_b = 2n^2 - \frac{b}{n+1}$.

We say that the element $s_\tau$ corresponds to the four vertices $v_\tau, v_{\tau+4n^2+1}, v_{\tau+8n^2+2},$ and $v_{\tau+12n^2+3}.$ Note that $s_1$ (which always exists) corresponds to the four vertices adjacent to $y_{ij}.$ Note also that $2n^2 - 1 < \ell''_a, \ell''_b, \ell'_b < 2n^2,$ since $a, b \in [n].$

The gadgets $G_{ij}$ are now connected to each other in a grid-like fashion (see Fig. 1b). That is, for $j \leq \kappa - 1$ we add a path $P_{ij}$ between $x^2_{ij}$ and $x^4_{ij+1}$ with $n + 2$ edges of length $\frac{1}{n+2}$ each. Analogously, for $i \leq \kappa - 1$ we introduce a path $P'_{ij}$...
between \( x_{i,j}^3 \) and \( x_{i+1,j}^1 \) that has \( n + 2 \) edges, each of length \( \frac{1}{n+2} \). Note that these paths all have length 1.

The resulting graph \( G_T \) forms an instance of \( k\text{-CENTER} \) with \( k = 5k^2 \). We claim that the instance \( I \) of \( G_T \) has a solution if and only if the optimum solution to \( k\text{-CENTER} \) on \( G_T \) has cost at most \( 2n^2 \). We note at this point that the reduction would still work when removing the vertices \( y_{i,j} \) and decreasing \( k \) to \( 4k^2 \). However, their existence will greatly simplify analysing the doubling dimension of \( G_T \) in Sect. 3.

### 2.2 A Solution to the \( G_T \) Instance Implies a \( k\text{-CENTER} \) Instance with Cost \( 2n^2 \)

Recall that we fixed an order of each set \( S_{i,j} \), so that each element \( s_t \in S_{i,j} \) corresponds to four equidistant vertices on cycle \( O_{i,j} \) with distance \( 4n^2 + 1 \) between consecutive such vertices on the cycle. If \( s_t \in S_{i,j} \) is in the solution to the \( G_T \) instance \( I \), let \( C_{i,j} = \{v_{r}, v_{r+4n^2+1}, v_{r+8n^2+2v_{r+12n^2+3}}, y_{i,j}\} \) contain the vertices of \( O_{i,j} \) corresponding to \( s_t \), in addition to \( y_{i,j} \). The solution to the \( k\text{-CENTER} \) instance \( G_T \) is given by the union \( \bigcup_{i,j \in [k]} C_{i,j} \), which consists of exactly \( 5k^2 \) centers in total.

Let us denote the set containing the four vertices of \( C_{i,j} \cap V(O_{i,j}) \) by \( C_{i,j}^O \) and note that each of these four vertices covers \( 4n^2 + 1 \) vertices of \( O_{i,j} \) with balls of radius \( 2n^2 \), as each edge of \( O_{i,j} \) has length 1. Since the distance between any pair of centers in \( C_{i,j} \) is at least \( 4n^2 + 1 \), these four sets of covered vertices are pairwise disjoint. Thus the total number of vertices covered by \( C_{i,j} \) on \( O_{i,j} \) is \( 16n^2 + 4 \), i.e., all vertices of the cycle \( O_{i,j} \) are covered. Recall that the lengths of the edges between the vertices \( x_{i,j}^1 \), \( x_{i,j}^2 \), \( x_{i,j}^3 \), and \( x_{i,j}^4 \) and the cycle \( O_{i,j} \) are \( \ell'_a, \ell'_b, \ell'_b < 2n^2 \). Hence the centers in \( C_{i,j} \) also cover \( x_{i,j}^1, x_{i,j}^2, x_{i,j}^3, \) and \( x_{i,j}^4 \) by balls of radius \( 2n^2 \).

Now consider a path connecting two neighbouring gadgets, e.g., \( P_{i,j} \) connecting \( x_{i,j}^2 \) and \( x_{i+1,j}^3 \). The center sets \( C_{i,j}^O \) and \( C_{i+1,j}^O \) contain vertices corresponding to the respective elements \( s \in S_{i,j} \) and \( s' \in S_{i+1,j} \) of the solution to the \( G_T \) instance. This means that if \( s = (a,b) \) and \( s' = (a',b') \) then \( b \leq b' \). Thus the closest centers of \( C_{i,j} \) and \( C_{i+1,j} \) are at distance \( \ell'_b + 1 + \ell'_b \) from each other, as \( P_{i,j} \) has length 1. From \( b \leq b' \) we get

\[
\ell'_b + 1 + \ell'_b = 2n^2 + \frac{b}{n+1} - 1 + 1 + 2n^2 - \frac{b'}{n+1} \leq 4n^2.
\]

Therefore all vertices of \( P_{i,j} \) are covered by the balls of radius \( 2n^2 \) around the two closest centers of \( C_{i,j} \) and \( C_{i+1,j} \). Analogously, we can also conclude that any path \( P'_{i,j} \) connecting some vertices \( x_{i,j}^1 \) and \( x_{i+1,j}^4 \) is covered, using the fact that if \( (a,b) \in S_{i,j} \) and \( (a',b') \in S_{i+1,j} \) are in the solution to the \( G_T \) instance then \( a \leq a' \).

Finally, the remaining center vertices in \( \bigcup_{i,j \in [k]} C_{i,j} \setminus C_{i,j}^O \) cover the additional vertex \( y_{i,j} \) in each gadget \( G_{i,j} \).
2.3 A k-Center Instance with Cost $2n^2$ Implies a Solution to the GT Instance

Each vertex $y_{ij}$ must be contained in any solution of cost at most $2n^2$, since the distance from $y_{ij}$ to any other vertex is more than $2n^2$. This already uses $\kappa^2$ of the available $5\kappa^2$ centers.

We now prove that in any solution to the $k$-Center instance $G_Z$ of cost at most $2n^2$, each cycle $O_{ij}$ must contain exactly four centers. Recall that $\ell_a, \ell'_a, \ell_b, \ell'_b > 2n^2 - 1$, that $y_{ij}$ is incident to four edges of length $2n^2 + 1$ each, and that each edge of $O_{ij}$ has length 1. Now consider the vertices $v_{4n^2+1}, v_{8n^2+2}, v_{12n^2+3}, v_{16n^2+4}$ of each of which is not connected by an edge to any vertex $x_{ij}^q$, where $q \in [4]$, nor to $y_{ij}$. Thus each of these four vertices must be covered by centers on the cycle $O_{ij}$ if the radius of each ball is at most $2n^2$. Furthermore, the distance between each pair of these four vertices is at least $4n^2 + 1$, which means that any solution of cost at most $2n^2$ needs at least four centers on $O_{ij}$ to cover these four vertices. Since there are $\kappa^2$ cycles and only $4\kappa^2$ remaining available centers, we proved that each cycle $O_{ij}$ contains exactly four centers, and apart from the $y_{ij}$ vertices no other centers exist in the graph $G_Z$.

Let $C_{ij}^O$ be the set of four centers contained in $O_{ij}$. As each center of $C_{ij}^O$ covers at most $4n^2 + 1$ vertices of $O_{ij}$, by balls of radius at most $2n^2$, to cover all $16n^2 + 4$ vertices of $O_{ij}$ these four centers must be equidistant with distance exactly $4n^2 + 1$ between consecutive centers on $O_{ij}$. Furthermore, since $\ell_a, \ell'_a, \ell_b, \ell'_b > 2n^2 - 1$ and each edge of $O_{ij}$ has length 1, to cover $x_{ij}^q$ for any $q \in [4]$ some center of $C_{ij}^O$ must lie on a vertex of $O_{ij}$ adjacent to $x_{ij}^q$. This means that the four centers of $C_{ij}^O$ are exactly those vertices $v_{r+(q-1)(4n^2+1)}$ corresponding to element $\ell_{r}^t$ of $S_{ij}$.

It remains to show that the elements corresponding to the centers in $\bigcup_{i,j \in [\kappa]} C_{ij}^O$ form a solution to the GT instance $I$. For this, consider two neighbouring gadgets $G_{ij}$ and $G_{ij+1}$, and let $(a, b) \in S_{ij}$ and $(a', b') \in S_{ij+1}$ be the respective elements corresponding to the center sets $C_{ij}^O$ and $C_{ij+1}^O$. Note that for any $b \in [n]$ we have $\ell_b \leq \ell_b' + 1$ and $\ell_b' \leq \ell_b + 1$. Since every edge of the cycles $O_{ij}$ and $O_{ij+1}$ has length 1, this means that the distance from the closest centers $v \in C_{ij}^O$ and $v' \in C_{ij+1}^O$ to $x_{ij}^2$ and $x_{ij+1}^4$, respectively, is determined by the edges of length $\ell_b$ and $\ell_b'$ incident to $v$ and $v'$, respectively. In particular, the distance between $v$ and $v'$ is $\ell_b + 1 + \ell_b'$, as the path $P_{ij}$ connecting $x_{ij}^2$ and $x_{ij+1}^4$ has length 1. Assume now that $b > b'$, which means that $b \geq b' + 1$ since $b$ and $b'$ are integer. Hence this distance is

$$\ell_b + 1 + \ell_b' = 2n^2 + \frac{b}{n + 1} - 1 + 1 + 2n^2 - \frac{b'}{n + 1} = 4n^2 + \frac{1}{n + 1}.$$

As the centers $v$ and $v'$ only cover vertices at distance at most $2n^2$ each, while the edges of the path $P_{ij}$ have length $\frac{1}{n+2} < \frac{1}{n+1}$, there must be some vertex of $P_{ij}$ that is not covered by the center set. However this contradicts the fact that the centers form a feasible solution with cost at most $2n^2$, and so $b \leq b'$.

An analogous argument can be made for neighbouring gadgets $G_{ij}$ and $G_{i+1,j}$, so that $a \leq a'$ for the elements $(a, b) \in S_{ij}$ and $(a', b') \in S_{i+1,j}$ corresponding to the centers in $C_{ij}^O$ and $C_{i+1,j}^O$, respectively. Thus a solution to $G_Z$ of cost at most $2n^2$ implies a solution to $I$. 

 Springer
3 Properties of the Constructed Graph

The reduction of Sect. 2 proves that the $k$-CENTER problem is W[1]-hard for parameter $k$, since the reduction can be done in polynomial time and $k$ is a function of $\kappa$. Since this function is quadratic, we can also conclude that, under ETH, there is no $f(k) \cdot n^{o(\sqrt{k})}$ time algorithm for $k$-CENTER. We will now show that the graph constructed in the reduction has various additional properties from which we will be able to conclude Theorem 1. First off, it is easy to see that any constructed graph $G_\mathcal{T}$ for an instance $\mathcal{T}$ of $GT_\leq$ is planar (cf. Fig. 1). We go on to prove that $G_\mathcal{T}$ has constant doubling dimension.

**Lemma 6** The graph $G_\mathcal{T}$ has doubling dimension at most $\log_2(324) \approx 8.34$ for $n \geq 2$.

**Proof** To bound the doubling dimension of the graph $G_\mathcal{T}$, consider the shortest-path metric on the vertex set $Y = \{ y_{i,j} \in V(G_\mathcal{T}) | i,j \in [\kappa] \}$ given by the distances between these vertices in $G_\mathcal{T}$. On an intuitive level, as these vertices are arranged in a grid-like fashion, the shortest-path metric on $Y$ approximates the $L_1$-metric. We consider a set of index pairs, for which the corresponding vertices in $Y$ roughly resemble a ball in the shortest-path metric on $Y$. That is, for any $a \in \mathbb{N}_0$ consider the set of index pairs $A_{i,j}(a) = \{ (i',j') \in [\kappa]^2 | |i - i'| + |j - j'| \leq a \}$, and let $V_{i,j}(a) \subseteq V(G_\mathcal{T})$ contain all vertices of gadgets $G_{i',j'}$ such that $(i',j') \in A_{i,j}(a)$ in addition to the vertices of paths of length 1 connecting these gadgets to each other and to any adjacent gadgets $G_{i'',j''}$ such that $(i'',j'') \notin A_{i,j}(a)$. We call the vertices $x_{p,q} \in V_{i,j}(a)$ such that $(i'',j'') \notin A_{i,j}(a)$, i.e., the endpoints of the latter paths of length 1, the boundary vertices of $V_{i,j}(a)$. We consider $y_{i,j}$ as the center of $V_{i,j}(a)$. We would like to determine the smallest radius of a ball around $y_{i,j}$ that contains all of $V_{i,j}(a)$, and the largest radius of a ball around $y_{i,j}$ that is entirely contained in $V_{i,j}(a)$. For this we need the following claim, which we will also reuse later.

**Claim 7** For any gadget $G_{i,j}$ and $q,q' \in [4]$ with $q \neq q'$, the distance between $x_{i,j}^q$ and $x_{i,j}^{q'}$ in $G_\mathcal{T}$ lies between $7n^2 - 1$ and $8n^2 + 2$.

**Proof** The distance between $x_{i,j}^q$ and $x_{i,j}^{q'}$ is less than $2(2n^2 + 2n^2 + 1) = 8n^2 + 2$, via the path passing through $y_{i,j}$ and the two vertices of $O_{i,j}$ adjacent to $y_{i,j}$, $x_{i,j}^q$, and $x_{i,j}^{q'}$. Note that the shortest path between $x_{i,j}^q$ and $x_{i,j}^{q'}$ inside the gadget $G_{i,j}$ does not necessarily pass through $y_{i,j}$, but may pass along the cycle $O_{i,j}$ instead. This is because the set $S_{i,j}$ of the $GT_\leq$ instance may contain up to $n^2$ elements, which would imply a direct edge from $x_{i,j}^q$ to $v_{n^2+(q-1)(4n^2+1)}$ on $O_{i,j}$. Thus we can give a lower bound of $2(2n^2 - 1) + 3n^2 + 1 = 7n^2 - 1$ for the distance between $x_{i,j}^q$ and $x_{i,j}^{q'}$ inside of $G_{i,j}$. This is also the shortest path between these vertices in $G_{\mathcal{T}}$, since any other path needs to pass through at least three gadgets. \[\square\]

We define the **circumradius** of $V_{i,j}(a)$ as the maximum distance from $y_{i,j}$ to any vertex inside of $V_{i,j}(a)$, while the **inradius** of $V_{i,j}(a)$ is the minimum distance from $y_{i,j}$...
to any vertex outside of $V_{ij}(a)$. Note that $V_{ij}(a) \subseteq B_{y_{ij}}(r)$ if $r$ is the circumradius, and $B_{y_{ij}}(r - \varepsilon) \subseteq V_{ij}(a)$ for any $\varepsilon > 0$ if $r$ is the inradius. Any shortest path from $y_{ij}$ to a vertex in $V_{ij}(a)$ passes through the gadget $G_{ij}$, at most $a$ additional gadgets $G_{i',j'}$ with $(i',j') \in A_{ij}(a)$, the paths of length 1 connecting these gadgets, and possibly one path of length 1 to reach a boundary vertex of $V_{ij}(a)$. Hence, by Claim 7, the circumradius of $V_{ij}(a)$ is less than $(8n^2 + 2)a + (a + 1) + 4n^2 + 1 = (8n^2 + 3)a + 4n^2 + 2$, since the distance from $y_{ij}$ to any $x_{ij}^q$ is less than $4n^2 + 1$. To reach any vertex outside of $V_{ij}(a)$ from $y_{ij}$ it is necessary to first reach $x_{ij}^q$ for some $q \in [4]$, then pass through $a$ gadgets $G_{i',j'}$ with $(i',j') \in A_{ij}(a)$, in addition to $a$ paths of length 1 connecting them and $G_{ij}$, and finally pass through another path of length 1 to reach a boundary vertex of $V_{ij}(a)$. From the boundary, a vertex not in $V_{ij}(a)$ can be reached on some cycle $O_{i',j'}$ with $(i'',j'') \notin A_{ij}(a)$. The distance from $y_{ij}$ to $x_{ij}^q$ is more than $4n^2$ and the distance from a boundary vertex $x_{i',j'}^q$ to any vertex of $O_{i',j'}$ is more than $2n^2 - 1$. Hence, by Claim 7, the inradius of $V_{ij}(a)$ is more than $(7n^2 - 1)a + (a + 1) + 4n^2 + (2n^2 - 1) = 7n^2a + 6n^2$.

Now consider any ball $B_v(2r)$ of radius $2r$ around some vertex $v$ in $G_\mathcal{I}$ for which we need to bound the number of balls of half the radius with which to cover $B_v(2r)$. Let $y_{ij}$ be the closest vertex of $Y$ to $v$. The distance between $y_{ij}$ and $v$ is at most $2(2n^2 + 1) = 4n^2 + 2$, whether $v$ lies on $O_{ij}$ or on one of the paths of length 1 connecting $G_{ij}$ with an adjacent gadget. Hence the ball $B_v(2r)$ is contained in a ball of radius $4n^2 + 2 + 2r$ around $y_{ij}$. The latter ball is in turn contained in the set $V_{ij}(a)$ centered at $y_{ij}$ if the ball’s radius is less than the inradius of $V_{ij}(a)$. This in particular happens if $4n^2 + 2 + 2r \leq 7n^2a + 6n^2$, which for instance is true if $a = \lceil \frac{2+2r-2n^2}{7n^2} \rceil$. Assume first that $r \geq 12n^2 + 5$, which implies that $a > 0$ and so $V_{ij}(a)$ is well-defined.

At the same time, any set $V_{i',j'}(a')$ is contained in a ball of radius $r$ around $y_{i',j'}$ if its circumradius is at most $r$, i.e., $(8n^2 + 3)a' + 4n^2 + 2 \leq r$. This is for instance true if $a' = \lfloor \frac{r-4n^2-2}{8n^2+3} \rfloor$. Note that $r \geq 12n^2 + 5$ means that $a' \geq 0$ and so $V_{i',j'}(a')$ is well-defined. We may cover all vertices of $A_{ij}(a)$ with $\lceil \frac{2a+1}{2a'+1} \rceil^2$ sets $A_{i',j'}(a')$, since in $Y$ these sets correspond to “squares rotated by 45 degrees” (i.e., balls in $L_1$) of diameter $2a + 1$ and $2a' + 1$, respectively. Thus we can cover $V_{ij}(a)$ with $\lceil \frac{2a+1}{2a'+1} \rceil^2$ sets $V_{i',j'}(a')$, i.e., we can cover a ball of radius $2r$ in $G_\mathcal{I}$ with

$$\left[ \frac{2a+1}{2a'+1} \right]^2 \leq \left[ \frac{2\left(\frac{2+2r-2n^2}{7n^2}\right) + 3}{2\left(\frac{r-4n^2-2}{8n^2+3}\right) - 1} \right]^2 = \left[ \frac{(8n^2 + 3)(4 + 4r + 17n^2)}{7n^2(2r - 7 - 16n^2)} \right]^2 \leq \left[ \frac{9(65r - 37)}{7(8r - 4)} \right]^2 \leq 121$$

balls of radius $r$, using that $r \geq 12n^2 + 5$ implies $n^2 \leq (r - 5)/12$.

Next consider the case when $2n^2 + 1 \leq r < 12n^2 + 5$. We know from above that $B_v(2r)$ is contained in $V_{ij}(a)$ if $a = \lceil \frac{2+2r-2n^2}{7n^2} \rceil$, which is well-defined as $r \geq 2n^2 + 1$ implies $a \geq 0$. Using $r < 12n^2 + 5$ and $n \geq 2$ we get $a \leq 4$. The set $V_{ij}(4)$ contains at most $(2 \cdot 4 + 1)^2 = 81$ gadgets. On each of the cycles $O_{i',j'}$ with $(i'',j'') \in A_{ij}(4)$ we

Springer
may choose the four vertices \( v_1, v_{4n^2+2}, v_{8n^2+3}, \) and \( v_{12n^2+4} \) adjacent to \( y_{ij} \) as centers for balls of radius \( r \). Note that as \( r \geq 2n^2 + 1 \), the vertex \( y_{ij} \), every vertex of the cycle \( O_{ij} \), and also every vertex on a path of length 1 adjacent to gadget \( G_{ij} \) is at distance at most \( r \) to one of these four vertices. Thus at most \( 4 \cdot 81 = 324 \) balls of half the radius are needed to cover all vertices of \( B_r(2r) \).

The next case we consider is \( n^2 - 1 \leq r < 2n^2 + 1 \). Again, \( B_r(2r) \) is contained in \( V_{ij}(a) \) if \( a = \frac{2+2r-2n^2}{7n^2} \), which is well-defined as \( r \geq n^2 - 1 \) implies \( a \geq 0 \). Using \( r < 2n^2 + 1 \) and \( n \geq 1 \) we obtain \( a \leq 1 \), which in turn means that the number of gadgets in \( V_{ij}(a) \) now is at most \( (2a + 1)^2 \leq 9 \). To cover a cycle \( O_{ij} \) with \( (i', j') \in A_{ij}(1) \), we may choose centers for balls of radius \( r \) equidistantly at every \( 2r \)-th vertex of \( O_{ij} \), as all edges of the cycle have length 1. We may lower bound \( |2r| \geq 2n^2 - 3 \) using \( r \geq n^2 - 1 \). Since every cycle contains \( 16n^2 + 4 \) vertices, the number of balls to cover a cycle is at most \( 16n^2 + 4/|2r| \leq 68/5 \leq 14 \), using the previous bound and \( n \geq 2 \). Hence at most \( 14 \cdot 9 = 126 \) balls of half the radius are needed to cover the cycles in \( B_r(2r) \). We can then cover the 9 vertices \( y_{ij} \) where \( (i', j') \in A_{ij}(1) \) and the \( 2 \cdot 9 + 2 \cdot 3 = 24 \) paths of length 1 contained in \( V_{ij}(a) \) with a ball of radius \( r \) each, as \( r \geq n^2 - 1 \geq 1 \) using \( n \geq 2 \). Hence a total of at most \( 9 + 24 + 126 = 159 \) balls of half the radius suffice to cover \( B_r(2r) \).

Finally, if \( r < n^2 - 1 \), then a ball \( B_r(2r) \) contains only a subpath of some cycle \( O_{ij} \), a subpath of a path of length 1 connecting two gadgets, or a single vertex \( y_{ij} \); since any edge connecting a cycle \( O_{ij} \) to \( y_{ij} \) or some \( x_{ij}^q \) has length more than \( 2n^2 - 1 > 2(n^2 - 1) > 2r \). In this case at most 3 balls of radius \( r \) suffice to cover all vertices of \( B_r(2r) \).

We next show that we can bound the parameters \( p \) and \( h \), i.e., the pathwidth and highway dimension of \( G_I \), linearly by \( k \) and \( \Theta(k^2) \), respectively. Note that the following lemma bounds the highway dimension in terms of \( k \), no matter how restrictive we make Definition 4 by increasing the constant \( c \).

**Lemma 8** For any constant \( c \) of Definition 4, the graph \( G_I \) has highway dimension at most \( O(k^2) \).

**Proof** For any scale \( r \in \mathbb{R}^+ \) and universal constant \( c \geq 4 \) we will define a hub set \( H_r \subseteq V \) hitting all shortest paths of length more than \( r \) in \( G_I \), such that \( |H_r \cap B_r(v)| = O(k^2) \) for any ball \( B_r(v) \) of radius \( cr \) in \( G_I \). This bounds the highway dimension to \( O(k^2) \) according to Definition 4.

Let \( X = \{ y_{ij}, x_{ij}^q \mid q \in \{4\} \text{ and } i, j \in [\kappa] \} \) so that it contains all vertices connecting gadgets \( G_{ij} \) to each other in addition to the vertices \( y_{ij} \). If \( r > 8n^2 + 2 \) then \( H_r = X \). Any shortest path containing only vertices of a cycle \( O_{ij} \) has length at most \( 8n^2 + 2 \), since the cycle has length \( 16n^2 + 4 \). Any (shortest) path that is a subpath of a path connecting two gadgets has length at most 1. Hence any shortest path of length more than \( 8n^2 + 2 \) must contain some vertex of \( X \). The total size of \( X \) is \( 5\kappa^2 \), and so any ball, no matter its radius, also contains at most this many hubs of \( H_r \).

If \( 1 \leq r \leq 8n^2 + 2 \) then any path of length more than \( r \) but not containing any vertex of \( X \) must lie on some cycle \( O_{ij} = (v_1, v_2, \ldots, v_{16n^2+4}, v_1) \). We define the set \( W_r = \{ v_{1+\lambda|r} \in V(O_{ij}) \mid \lambda \in \mathbb{N}_0 \} \), i.e., it contains every \( r \)-th vertex on the cycle.
after rounding down. This means that every path on \( O_{ij} \) of length more than \( r \) contains a vertex of \( W_{ij} \). Thus for these values of \( r \) we set \( H_r = X \cup \bigcup_{i \in \mathbb{N}} W_{ij} \). Any ball \( B_{cr}(v) \) of radius \( cr \) contains \( O(c) \) hubs of any \( W_{ij} \). By Claim 7, the distance between any pair of the four vertices \( x_{ij}^q \), where \( q \in [4] \), that connect a gadget \( G_{ij} \) to other gadgets, is more than \( 7n^2 - 1 \). This means that \( B_{cr}(v) \) can only intersect \( O(c^2) \) gadgets, since \( cr \leq c(8n^2 + 2) \leq 2c(7n^2 - 1) \) if \( n \geq 1 \) and the gadgets are connected in a grid-like fashion. Hence the ball \( B_{cr}(v) \) only contains \( O(c) \) hubs for each of the \( O(c^2) \) sets \( W_{ij} \) for which \( B_{cr}(v) \) intersect the respective gadget \( G_{ij} \). At the same time each gadget contains only 5 vertices of \( X \). Thus if \( c \) is a constant, then the number of hubs of \( H_r \) in \( B_{cr}(v) \) is constant.

If \( r < 1 \), a path of length more than \( r \) may be a subpath of a path connecting two gadgets. Recall that the paths \( P_{ij} \) connecting \( x_{ij}^2 \) and \( x_{ij+1}^4 \) for \( j \leq \kappa - 1 \), and the paths \( P_{ij} \) connecting \( x_{ij}^1 \) and \( x_{ij+1}^3 \) for \( i \leq \kappa - 1 \), consist of \( n + 2 \) edges of length \( \frac{1}{n+2} \) each. If \( P_{ij} = (u_0, u_1, \ldots, u_{n+2}) \), we define the sequence \( U_{ij} = \{ u_{\lambda | (r+2n)} \in V(P_{ij}) | \lambda \in \mathbb{N}_0 \} \), and if \( P_{ij} = (u_0, u_1, \ldots, u_{n+2}) \), we define the sequence \( \tilde{U}_{ij} = \{ u_{\lambda | (r+2n)} \in V(P_{ij}) | \lambda \in \mathbb{N}_0 \} \), i.e., these sequences contain vertices of consecutive distance \( r \) on the respective paths, after rounding down. Now let \( H_r = \bigcup_{i \in \mathbb{N}} X(V(G_{ij}) \cup \bigcup_{i \in \mathbb{N}} \bigcup_{i \in [\kappa]} U_{ij} \cup \bigcup_{i \in [\kappa]} \tilde{U}_{ij} \) so that every path of length more than \( r \) contains a hub of \( H_r \). Any ball \( B_{cr}(v) \) of radius \( cr < c \) intersects only \( O(c^2) \) gadgets \( G_{ij} \), as observed above. As the edges of a cycle \( O_{ij} \) have length 1, the ball \( B \) contains only \( O(c) \) vertices of \( O_{ij} \). Thus \( B_{cr}(v) \) contains \( O(c) \) hubs of \( V(G_{ij}) \cup \tilde{U}_{ij} \cup \tilde{U}_{ij} \) for each of the \( O(c^2) \) gadgets \( G_{ij} \) it intersects. For constant \( c \), this proves the claim.

\[ \square \]

**Lemma 9** The graph \( G_X \) has pathwidth at most \( \kappa + O(1) \).

**Proof** We construct a path decomposition of \( G_X \) using bags of size \( \kappa + O(1) \). For each \( i, j \in [\kappa] \) we define the sets \( X_{ij}^2 = \{ x_{ij}^2 | i' \in [i] \} \) and \( X_{ij}^4 = \{ x_{ij}^4 | i' \in [\kappa] \setminus [i-1] \} \) and let \( K_{ij} = \{ y_{ij}, x_{ij}^1, x_{ij}^3 \} \cup X_{ij}^2 \cup X_{ij}^4 \) be a bag. Intuitively, these bags decompose the graph \( G_T \) “from left to right” according to Fig. 1. More precisely, using some additional intermediate bags, the constructed path decomposition will arrange these bags on a path with start vertex \( K_{1,1} \), such that traversing the path will consecutively move from \( K_{ij} \) to \( K_{i+1,j} \) for each \( 1 \leq i \leq \kappa - 1 \) and \( 1 \leq j \leq \kappa \), and from \( K_{\kappa,j} \) to \( K_{1,j+1} \) for each \( 1 \leq j \leq \kappa - 1 \).

To define the intermediate bags, consider a bag \( K_{ij} \) and note that the three connected components left after removing all vertices of \( K_{ij} \) from \( G_T \) are (a) the cycle \( O_{ij} \) (b) the subgraph \( L_{ij} \) “to the left of” \( K_{ij} \) induced by all gadgets \( G_{ij}^r \) and paths \( P_{r,j}^r, P_{r,j}^r \) for which \( j' \leq j - 1 \) and \( i' \leq \kappa \), but also the gadgets \( G_{ij}^r \) and paths \( P_{r,j}^r, P_{r,j}^r \) for which \( j' \geq j + 1 \) and \( i' \leq \kappa \), and finally (c) the subgraph \( R_{ij} \) “to the right of” \( K_{ij} \) induced by all gadgets \( G_{ij}^r \) and paths \( P_{r,j}^r, P_{r,j}^r \) for which \( j' = j \) and \( i' \leq 1 + 1 \), or \( j' \geq j + 1 \) and \( i' \leq \kappa \), but also the paths \( P_{r,j}^r, P_{r,j}^r \) where \( i' \leq i \) and the path \( P_{r,j}^r \).

For any \( i \leq \kappa - 1 \), removing the union \( K_{ij} \cup K_{i+1,j} \) from \( G_X \) leaves \( L_{ij}, R_{i+1,j}, O_{ij}, O_{i+1,j} \), and the path \( P_{ij}^r \) connecting the gadgets \( G_{ij} \) and \( G_{i+1,j} \). The intermediate bags connecting \( K_{ij} \) and \( K_{i+1,j} \) for \( 1 \leq \kappa - 1 \) on the path decomposition will first cover \( O_{ij} \) and then \( P_{ij}^r \) if \( O_{ij} = (v_1, v_2, \ldots, v_{16n^2+4}, v_1) \), we define a sequence of bags

\[ \square \]
\( K_{ij}^\tau = K_{ij} \cup \{ v_1, v_\tau, v_{\tau+1} \} \) where \( \tau \in [16n^2 + 3] \), and if \( P'_{ij} = (u_0, u_1, \ldots, u_{n+2}) \) where
\[
ru_{ij} = x^3_{ij} \quad \text{and} \quad u_{n+2} = x^4_{i+1,j},
\]
then we define a sequence of bags \( J_{ij}^\tau = K_{ij} \cup \{ u_{\tau-1}, u_\tau \} \) for \( \tau \in [n+2] \). Note that for every edge \( e \) of \( O_{ij} \) or \( P'_{ij} \) there is a bag containing the vertices of \( e \). Moreover, for every other edge \( e \) of gadget \( G_{ij} \) connecting \( O_{ij} \) to \( x^q_{ij} \) for \( q \in [4] \) or to \( y_{ij} \) there also is a bag \( K_{ij}^\tau \) containing the vertices of \( e \). Now, the path decomposition contains a subpath between the vertices corresponding to \( K_{ij} \) and \( K_{i+1,j} \), which starting from \( K_{ij} \) first traverses vertices for \( K_{ij}^1 \) with increasing index \( \tau \), then connects \( K_{ij}^{16n^2+3} \) to \( J_{ij}^1 \), then traverses through \( J_{ij}^1 \) with increasing \( \tau \), and finally connects \( J_{ij}^{n+2} \) to \( K_{i+1,j} \). That is, the sequence of bags defined by the subpath is
\[
\left( K_{ij}, K_{ij}^1, K_{ij}^2, \ldots, K_{ij}^{16n^2+3}, J_{ij}^1, J_{ij}^2, \ldots, J_{ij}^{n+2}, K_{i+1,j} \right).
\]

To connect \( K_{k,j} \) to \( K_{1,j+1} \) for some \( j \leq k - 1 \), we define additional bags \( K_{ij}' = X^2_{ij} \cup X^4_{i+1,j} \). Starting from \( K_{k,j} \) and using intermediate bags, the path decomposition will traverse the bags \( K_{ij}' \) with decreasing index \( i \) until reaching \( K_{1,j+1} \).

We first describe the bags of the path decomposition connecting \( K_{k,j} \) to the first additional bag \( K_{k,j}' \). Defining the intermediate bags is similar to above. For any \( i \in [\kappa] \), removing the vertices of \( K_{ij}' \) from \( G_z \) leaves three connected components of which one is \( P_{ij} \) connecting the respective gadgets \( G_{ij} \) and \( G_{i+1,j} \), one is a component \( L_{ij}' \), which is \( L_{1,j+1} \) without the paths \( P_{ij} \) where \( i' \leq i \), and one is a component \( R_{ij}' \), which is \( R_{k,j} \) without the paths \( P_{ij} \) where \( i' \geq i \). If \( P_{ij} = (u_0, u_1, \ldots, u_{n+2}) \) where \( u_0 = x^2_{ij} \) and \( u_{n+2} = x^4_{i+1,j} \), we define a sequence of bags \( I^\tau_{ij} = K_{ij}' \cup \{ u_{\tau-1}, u_\tau \} \) for \( \tau \in [n+2] \). Note that for every edge of \( P_{ij} \) there is a bag \( I^\tau_{ij} \) containing its vertices. Now, the path decomposition contains a subpath connecting vertices corresponding to \( K_{k,j} \) and \( K_{k,j}' \) which starting from \( K_{k,j} \) moves to \( K_{k,j}' \) with increasing index \( \tau \) to cover \( O_{k,j} \) and then connects \( K_{ij}^{16n^2+3} \) to \( I_{k,j}^1 \). It then traverses through \( I_{k,j}^\tau \) with increasing index \( \tau \) to cover \( K_{k,j} \), after which it moves on to \( K_{k,j}' \). That is, the sequence of bags defined by the subpath is
\[
\left( K_{k,j}, K_{k,j}^1, K_{k,j}^2, \ldots, K_{k,j}^{16n^2+3}, I_{k,j}^1, I_{k,j}^2, \ldots, I_{k,j}^{n+2}, K_{k,j}' \right).
\]

For any \( i \leq k - 1 \) the path decomposition contains a subpath connecting \( K_{i+1,j} \) to \( K_{ij}' \) covering \( P_{ij} \) via the bags \( I_{ij}^\tau \) with increasing index \( \tau \). The sequence defined by this subpath is
\[
\left( K_{i+1,j}^1, I_{ij}^1, I_{ij}^2, \ldots, I_{ij}^{n+2}, K_{ij}' \right).
\]

The last additional bag \( K_{ij}' \) is connected directly to \( K_{1,j+1} \) on the path decomposition.

Finally, when at \( K_{k,k} \) the path decomposition only needs to cover \( O_{k,k} \) to finish, i.e., to make sure that every vertex of \( G_z \) is contained in some bag. This can be done
using the sequence $K^\tau_{k,k}$ with increasing index $\tau$, as above. That is, the sequence of 
bags defined by the final subpath of the path decomposition is 

$$
\left( K^1_{k,k}, K^2_{k,k}, \ldots, K^{16n^2+3}_{k,k} \right)
$$

As argued above, for every edge of $G/T$ there is a bag containing its vertices. To 
argue that all bags containing some vertex of $G/T$ form a subpath of the path decom-
position, note that for intermediate bags $K^i_{i,j}$, $J^i_{i,j}$, and $I^i_{i,j}$ we have $K^i_{i,j}, J^i_{i,j} \supseteq K_{i,j}$ and 
$I^i_{i,j} \supseteq K'_{i,j}$. Also note that every vertex of $X = \{y_{i,j}, x^q_{i,j} | q \in [4] \text{ and } i,j \in [k] \}$ lies in 
some bag $K_{i,j}$ or $K'_{i,j}$. Let $x \in X$ be a vertex that appears in a bag $B$ but not in bag $B'$,
and assume first that $B$ comes before $B'$ in the sequence defined by the path decom-
position. Since the path decomposition traverses $G/T$ “from left to right”, this means 
that $x$ lies in the set $L_{i,j}$ of some bag $K_{i,j}$, or the set $L'_{i,j}$ of some bag $K'_{i,j}$, for which $B' \supseteq K_{i,j}$ or $B' \supseteq K'_{i,j}$, respectively. Similarly, if $B'$ comes before $B$ in the sequence, 
then $x$ lies in the set $R_{i,j}$ of some bag $K_{i,j}$, or the set $R'_{i,j}$ of some bag $K'_{i,j}$, for which $B' \supseteq K_{i,j}$ or $B' \supseteq K'_{i,j}$, respectively. Now observe that $L_{i,j} \subseteq L'_{i,j}$ and $R_{i,j} \supseteq R'_{i,j}$ for 
any $i < i'$ where $j = j'$ but also for any $j < j'$, while for bags $K'_{i,j}$, for any $j$ and $i > i'$ we 
have $L_{k,j} \subseteq L'_{k,j} \subseteq L'_{r,j}$ and $R_{r,j} \supseteq R'_{r,j} \supseteq R_{i,j}$. This means that, by definition of the 
sequence of bags along the path decomposition, if $B$ comes before $B'$ the vertex $x$ 
cannot appear in any bag after $B'$ either, while if $B'$ comes before $B$ then $x$ cannot 
appear in any bag before $B'$ either. As a consequence, the bags containing any $x \in X$ 
must form a subpath of the path decomposition.

It remains to argue about vertices not in $X$. Note that these only occur in inter-
mediate bags $K^i_{i,j}$, $J^i_{i,j}$, and $I^i_{i,j}$ on some cycle of a gadget or a path connecting gadgets. 
Furthermore, the vertices of a cycle $O_{i,j}$ only occur in the bags $K^i_{i,j}$, vertices of paths 
$P^i_{i,j}$ (except the endpoints which lie in $X$) only occur in the bags $J^i_{i,j}$, and vertices of 
paths $P_{i,j}$ (except the endpoints which lie in $X$) only occur in the bags $I^i_{i,j}$. It is thus 
easy to see from the definition of the sequences of bags above, that any vertex not in $X$ 
only lies in bags that form a subpath of the path decomposition.

Note that each bag contains $\kappa + O(1)$ vertices, which concludes the proof.

The reduction given in Sect. 2 together with Lemmas 6, 8, and 9 imply Theorem 1, since the $G_T \leq$ problem is $W[1]$-hard [9] for parameter $\kappa$, and we may assume 
w.l.o.g. that $n \geq 2$. Moreover $\kappa = \Theta(\sqrt{k})$ and, under ETH, $G_T \leq$ has no $f(\kappa) \cdot n^{O(\kappa)}$ 
time algorithm [9] for any computable function $f$.

### 4 An Algorithm for Low Doubling Metrics

In this section we give a simple algorithm that generalizes one of Agarwal and Pro-
copiuc [4], which for $D$-dimensional $L_q$ metrics computes a $(1 + \varepsilon)$-approximation in 
time $f(\varepsilon, k, D) \cdot n^{O(1)}$. In particular, any such metric has doubling dimension $O(D)$. 
Here we assume that the input metric has doubling dimension $d$. A fundamental 
observation about metrics of bounded doubling dimension is the following, which

\[ \text{Springer} \]
can be proved by a simple recursive application of Definition 2. Here the aspect ratio of a set \( Y \subseteq X \) is the diameter of \( Y \) divided by the minimum distance between any two points of \( Y \).

**Lemma 10** [19] Let \((X, \text{dist})\) be a metric with doubling dimension \( d \) and \( Y \subseteq X \) be a subset with aspect ratio \( \alpha \). Then \(|Y| \leq 2^d[\log_2 \alpha] \).

To compute a \((1 + \varepsilon)\)-approximation to \( k\text{-CENTER} \) given a graph \( G \) with vertex set \( V \), we first compute its shortest-path metric \((V, \text{dist})\). We then compute several \textit{nets} of this metric, which are defined as follows.

**Definition 11** For a metric \((X, \text{dist})\), a subset \( Y \subseteq X \) is called a \( \delta \)-\textit{cover} if for every \( u \in X \) there is a \( v \in Y \) such that \( \text{dist}(u, v) \leq \delta \). A \( \delta \)-\textit{net} is a \( \delta \)-cover with the additional property that \( \text{dist}(u, v) > \delta \) for all distinct points \( u, v \in Y \).

Note that a \( \delta \)-net can be computed greedily in polynomial time. The first step of our algorithm is to guess the optimum cost \( \rho \) by trying each of the \({n \choose k}\) possible values. For each guess we compute an \( \varepsilon \delta \)-net \( Y \subseteq V \). We know that the metric \((V, \text{dist})\) can be covered by \( k \) balls of diameter \( 2\rho \) each, which means that the aspect ratio of \( Y \) inside of each ball is at most \( 4/\varepsilon \). Thus by Lemma 10, each ball contains \( 1/\varepsilon^\Omega(d) \) vertices of \( Y \), and so \(|Y| \leq k/\varepsilon^\Omega(d) \).

An optimum \( k\text{-CENTER} \) solution \( C \subseteq Y \) for \((Y, \text{dist})\) can be computed by brute force in \( n^\Omega(1) = k^\Omega(1) \) steps. Since every center of the optimum solution \( C^* \subseteq V \) of the input graph has a net point of \( Y \) at distance at most \( \varepsilon \delta \), there exists a \( k\text{-CENTER} \) solution in \( Y \) of cost at most \((1 + \varepsilon/2)\rho \), given that \( \rho \) is the optimum cost. The computed center set \( C \subseteq Y \) thus also has cost at most \((1 + \varepsilon/2)\rho \). Therefore \( C \) covers all of \( V \) with balls of radius \((1 + \varepsilon)\rho \), since every vertex of \( V \) is at distance \( \varepsilon \delta \) from some vertex of \( Y \). Thus \( C \) is a \((1 + \varepsilon)\)-approximation of the input graph. Considering the guessed values of \( \rho \) in increasing order, outputting the first computed solution with cost at most \((1 + \varepsilon)\rho \) gives the algorithm of Theorem 3.

**Acknowledgements** We would like to thank the anonymous reviewers, who greatly helped to improve the quality of this manuscript.

**Funding** The first author is supported by the Czech Science Foundation GAČR (Grant #19-27871X), and by the Center for Foundations of Modern Computer Science (Charles Univ. Project UNCE/SCI/004). The second author is supported by ERC Consolidator Grant SYSTEMATICGRAPH (No. 725978). An extended abstract of this paper previously appeared at SWAT 2018 [16].

**References**


Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
Polynomial Time Approximation Schemes for Clustering in Low Highway Dimension Graphs

Andreas Emil Feldmann
Charles University, Prague, Czech Republic
feldmann.a.e@gmail.com

David Saulpic
LIP6, Sorbonne Université, Paris, France
david.saulpic@lip6.fr

Abstract
We study clustering problems such as $k$-Median, $k$-Means, and Facility Location in graphs of low highway dimension, which is a graph parameter modeling transportation networks. It was previously shown that approximation schemes for these problems exist, which either run in quasi-polynomial time (assuming constant highway dimension) [Feldmann et al. SICOMP 2018] or run in FPT time (parameterized by the number of clusters $k$, the highway dimension, and the approximation factor) [Becker et al. ESA 2018, Braverman et al. 2020]. In this paper we show that a polynomial-time approximation scheme (PTAS) exists (assuming constant highway dimension). We also show that the considered problems are NP-hard on graphs of highway dimension 1.

2012 ACM Subject Classification Theory of computation → Facility location and clustering

Keywords and phrases Approximation Scheme, Clustering, Highway Dimension

Digital Object Identifier 10.4230/LIPIcs.ESA.2020.46


Funding Andreas Emil Feldmann: Supported by the Czech Science Foundation GAČR (grant #19-27871X), and by the Center for Foundations of Modern Computer Science (Charles Univ. project UNCE/SCI/004).

Acknowledgements We thank Vincent Cohen-Addad for helpful discussions.

1 Introduction

Clustering is a standard optimization task that seeks a “good” partition of a metric space, such that two points that are “close” should be in the same part. A good clustering of a dataset allows to retrieve and exploit data, and is therefore a common routine in data analysis. The underlying data can come from various sources and represent many different objects. In particular, it is often interesting to cluster geographic data. In that case, the metric space can be given by a transportation network, which can be modeled by graphs with low highway dimension.

In this article, we study some popular clustering objectives, namely Facility Location, $k$-Median, and $k$-Means, in graphs with constant highway dimension. The two latter problems seek to find a set $S$ of $k$ points called centers in a metric $(V, \text{dist})$ that minimizes $\sum_{v \in V} (\min_{f \in S} \text{dist}(v, f))^p$, with $p = 1$ for $k$-Median and $p = 2$ for $k$-Means. The objective for Facility Location is slightly different: each point $f$ of the metric space has an opening cost $w_f$, and the goal is to find a set $S$ that minimizes $\sum_{f \in S} w_f + \sum_{v \in V} \min_{f \in S} \text{dist}(v, f)$. These problems are APX-hard in general metric spaces [4].

To bypass the hardness of approximation known for these problems, researchers have considered low dimensional input, such as Euclidean spaces of fixed dimension, metrics with bounded doubling dimension, or with bounded genus. Many algorithmic tools were developed...
PTASs for Clustering in Low Highway Dimension Graphs

for that purpose: in their seminal work, Arora et al. [3] gave the first polynomial time approximation scheme (PTAS) for \(k\)-\text{Median} in \(\mathbb{R}^2\), which generalizes to a quasi-polynomial time approximation scheme (QPTAS) in \(\mathbb{R}^d\) for fixed \(d\). This result was generalized by Talwar [20], who gave a QPTAS for metrics with bounded doubling dimension, and more recently by Cohen-Addad et al. [10], who gave a near-linear time approximation scheme.

In this work we focus on transportation networks, for which it can be argued that metric spaces with bounded doubling dimension are not a suitable model: for instance, hub-and-spoke networks seen in air traffic networks do not have low doubling dimension. Therefore we study graphs with constant highway dimension, which formalize structural properties of such networks. The following definition is taken from Feldmann et al. [14]. Here the ball \(\beta_v(r)\) of radius \(r\) around \(v\) is the set of all vertices at distance at most \(r\) from \(v\).

\begin{definition}
The highway dimension of a graph \(G\) is the smallest integer \(h\) such that, for some universal constant \(c > 4\), for every \(r \in \mathbb{R}^+\), and every ball \(\beta_v(cr)\) of radius \(cr\), there are at most \(h\) vertices in \(\beta_v(cr)\) hitting all shortest paths of length more than \(r\) that lie in \(\beta_v(cr)\).
\end{definition}

For this class of graphs, the only known approximation algorithms for clustering that compute \((1 + \varepsilon)\)-approximations for any \(\varepsilon > 0\) either run in quasi-polynomial time, i.e., QPTASs [14], or with runtime \(f(h, k, \varepsilon) \cdot n\) for some exponential function \(f\), i.e., parameterized approximation schemes [6, 8]. Thus an open problem is to identify polynomial-time approximation schemes (PTASs) for clustering in graphs of constant highway dimension.

1.1 Our results

Our main result is a PTAS for clustering problems on graphs of constant highway dimension. For convenience, we define slightly more general problems than those stated above. The \(k\)-\text{Clustering} problem is defined as follows. An instance \(I\) consists of a metric \((V, \text{dist})\), a set of facilities (or centers) \(F \subseteq V\), and a demand function \(\chi : V \rightarrow \mathbb{N}_0\). The goal is to find a set \(S \subseteq F\) with \(|S| \leq k\) minimizing \(\sum_{v \in V} \chi(v) \cdot \min_{f \in S} \text{dist}(v, f)\). We call all vertices \(v \in V\) with \(\chi(v) > 0\) the clients of \(I\). \(k\)-\text{Median} and \(k\)-\text{Means} are special cases of \(k\)-\text{Clustering}, where \(q = 1\) and \(q = 2\).

The input to the \text{Facility Location} problem is the same as for \(k\)-\text{Clustering}, but additionally each facility \(f \in F\) has an opening cost \(w_f \in \mathbb{R}^+\). The goal is to find a set \(S \subseteq F\) minimizing \(\sum_{f \in S} w_f + \sum_{v \in V} \chi(v) \cdot \min_{f \in S} \text{dist}(v, f)\). \text{Facility Location} is a special case of \text{Facility Location}, where \(q = 1\).

Our main theorem is the following, where \(X = \max_{v \in V} \chi_I(v)\) is the largest demand (note that for \(k\)-\text{Median}, \(k\)-\text{Means}, or \text{Facility Location} we typically have \(X = 1\)).

\begin{theorem}
For any \(\varepsilon > 0\), a \((1 + \varepsilon)\)-approximation for \(k\)-\text{Clustering} and \text{Facility Location} can be computed in \((nX)^{O(q\varepsilon^2)}\) time on graphs of highway dimension \(h\).
\end{theorem}

In particular, this algorithm is much faster than the quasi-polynomial time approximation scheme of Feldmann et al. [14] for \(k\)-\text{Median} or \text{Facility Location}. The runtime of our algorithm also significantly improves over the exponential dependence on \(k\) in the approximation schemes of Becker et al. [6], Braverman et al. [8] for \(k\)-\text{Median}.

It has so far been open whether these clustering problems are NP-hard on graphs of constant highway dimension. We complement our main theorem by showing that they are NP-hard even for the smallest possible highway dimension. This answers an open problem given in [14]. Here the \emph{uniform} \text{Facility Location} problem has unit opening costs for all facilities.
Theorem 3. The \( k\)-\textsc{Clustering} and uniform \textsc{Facility Location} problems are NP-hard on graphs of highway dimension 1.

1.2 Related work

On clustering problems. The problems we focus on in this article are known to be APX-hard, even in Euclidean spaces (see e.g. [4]). In general metric spaces, the current best polynomial-time algorithm for \textsc{Facility Location} achieves a 1.488-approximation [19], while the best approximation factor is 2.67 for \textsc{k-Median} ([9]) and 6.357 for \textsc{k-Means} [2].

When restricting the class of graphs, a near-linear time approximation scheme for doubling metrics was developed in [10]; we will discuss the close relations between our work and this one in Section 1.3. Local search techniques also yield a PTAS in minor-free graphs or with bounded doubling dimension [11, 15], and a \( \Theta(q)\)-approximation for the \( k\)-\textsc{Clustering} problem in general metric spaces [17].

Another technique for dealing with clustering problems is to compute coresets, a compressed representation of the input. An \( \varepsilon\)-coreset is a weighted set of points such that for every set of centers, the cost for the original set of points is within a \( (1 + \varepsilon)\)-factor of the cost for the coreset. Braverman et al. [8] recently proved that graphs with highway dimension \( h \) admit coreset of size \( \tilde{O}\left((k + h)^{O(1/\varepsilon)}\right)\). This enables to compute a \( (1 + \varepsilon)\)-approximation by enumerating all possible solutions of the coreset. However, this coreset does not have small highway dimension,\(^1\) and thus cannot be used to boost our algorithms.

On highway dimension. The highway dimension was originally defined by Abraham et al. [1], who specifically chose balls of radius 4\( r \) in the Definition 1. Since the original definition in [1], several other definitions have been proposed. In particular, Feldmann et al. [14] proved that when choosing a radius \( cr \) in Definition 1 for any constant \( c \) strictly larger than 4, it is possible to exploit the structure of graphs with constant highway dimension in order to obtain a QPTAS for problems such as TSP, \textsc{Facility Location}, and \textsc{Steiner Tree}. As Abraham et al. [1] point out, the choice of the constant is somewhat arbitrary, and we use the above definition so that we may exploit the structural insights of [14] for our algorithm. These structural properties were also leveraged by Becker et al. [6] who gave a PTAS for the \textsc{Bounded-Capacity Vehicle Routing} problem, and a parameterized approximation scheme for the \textsc{k-Center} problem (which is essentially \( k\)-\textsc{Clustering} with \( q = \infty \)) and \( k\)-\textsc{Median}. In the lower bound side, Disser et al. [12] showed that \textsc{Steiner Tree} and TSP are weakly NP-hard even when the highway dimension is 1, i.e., each of them is NP-hard but an FPTAS exists for graphs of highway dimension 1.

It is worth mentioning that further definitions of the highway dimension exist (for a detailed discussion see [7, 14]). In particular, for a more general definition of the highway dimension than the one of Definition 1, Feldmann [13] gave a parameterized \( 3/2 \)-approximation algorithm with runtime \( 2^{O(kh \log h)} n^{O(1)} \) for \textsc{k-Center}.

1.3 Our techniques

To obtain Theorem 2, we rely on the framework recently developed by Cohen-Addad et al. [10] for doubling metrics. More precisely, they show that the split-tree decomposition of Talwar [20] has some interesting properties, and exploit them to design their algorithm.

\(^1\) Indeed, a subset of a metric with small highway dimension does not necessarily have small highway dimension as well: think of a star metric on which the center is removed.
Our main contribution is to provide a decomposition with similar properties in graphs with constant highway dimension. This is done relying on some structural properties of such graphs presented by Feldmann et al. [14]. We start by giving the outline of the algorithm from [10], and then explain how to carry the results over to the highway dimension setting.

**On doubling metrics.** The starting point of many approximation algorithms for doubling metrics is a decomposition of the metric, as presented in the following lemma. A hierarchical decomposition $D$ of a metric $(V, \text{dist})$ is a set of partitions $\mathcal{B}_0, \mathcal{B}_1, \ldots, \mathcal{B}_\lambda$, where $\mathcal{B}_i$ refines $\mathcal{B}_{i+1}$, i.e., every part $B \in \mathcal{B}_i$ is contained in some part of $\mathcal{B}_{i+1}$. Moreover, in $\mathcal{B}_0$ every part contains a singleton vertex, while $\mathcal{B}_\lambda$ contains only one part, namely $V$. For a point $v \in V$ and a radius $r > 0$, we say that the ball $\beta_v(r)$ is cut at level $i$ if $i$ is the largest integer for which the ball $\beta_v(r)$ is not contained in a single part of $\mathcal{B}_i$. For any subset $W \subseteq V$ we define $\lambda(W) = \lceil \log_2 \text{diam}(W) \rceil$.

Lemma 4 (Reformulation of [20, 5] as found in [10]). For any metric $(V, \text{dist})$ of doubling dimension $d$ and any $\rho > 0$, there exists a polynomial-time computable randomized hierarchical decomposition $D = \{\mathcal{B}_0, \ldots, \mathcal{B}_\lambda(V)\}$ such that:
1. **Scaling probability:** for any $v \in V$, radius $r$, and level $i$, we have
   \[ \Pr[D \text{ cuts } \beta_v(r) \text{ at level } i] \leq 2^{O(d)} \cdot r^2. \]
2. **Portal set:** every part $B \in \mathcal{B}_i$ where $\mathcal{B}_i \in D$ comes with a set of portals $P_B \subseteq B$ that is
   a. **concise:** the size of the portal set is bounded by $|P_B| \leq 1/\rho^2$, and
   b. **precise:** for every node $u \in B$ there is a portal $p \in P_B$ with $\text{dist}(u, p) \leq \rho^2i+1$.

We sketch briefly the standard use of this decomposition. For clustering problems, one can show that there exists a portal-respecting solution with near-optimal cost (see Talwar [20]). In this structured solution, each client connects to a facility via a portal-respecting path that enters and leaves any part $B$ of $D$ only through a node of the portal set $P_B$. Those portals therefore act as separators of the metric. A standard dynamic program approach can then compute the best portal respecting solution.

To ensure that there is a portal-respecting solution with near-optimal cost, one uses the preciseness property of the portal set: the distortion of connecting a client $c$ with a facility $f$ through portals instead of directly is bounded as follows. Let $i$ be the level at which $D$ cuts $c$ and $f$, meaning that $i$ is the maximum integer for which $c$ and $f$ lie in different parts of $\mathcal{B}_i$. At every level $j \leq i$, the distortion incurred by using portals is $\rho^{2j}$. Hence the total distortion is $\sum_{j \leq i} \rho^{2j} = \rho^{2i+1}$. Now, property (1) of the decomposition ensures that $c$ and $f$ are cut at level $i$ with probability $O(\text{dist}(c, f)/2i)$. Hence combining those two bounds over all levels ensures that, in expectation, the distortion between $c$ and $f$ is $O(\text{dist}(c, f) \cdot \rho \lambda(V))$. Since $\lambda(V) = O(\log n)$, choosing $\rho = \epsilon \log n$ gives a distortion of $O(\epsilon \text{dist}(c, f))$. Summing over all clients proves that there exists a near-optimal portal-respecting solution.

The issue with this approach is that the number of needed portals is $O(\log^d n)$, and the dynamic program has a runtime that is exponential in this number. Thus the time complexity is quasipolynomial. The novelty of [10] is to show how to reduce the number of portals to a constant. The idea is to reduce the number of levels on which a client can be cut from its facility.

---

2 We remark that in [10] the preciseness of Lemma 4 was expressed akin to the weaker property found in Lemma 5, which however would not lead to a near-linear time approximation scheme as claimed in [10], but rather a PTAS as shown in this work. This can however easily be alleviated for [10] by using the stronger preciseness as stated here in Lemma 4.
For this, they present a processing step of the instance, that helps deal with clients cut from their facility at a high level. Roughly speaking, their algorithm computes a constant factor approximation $L$, and a client $c$ is called badly-cut if $D$ cuts it from its closest facility of $L$ at a level larger than $\log(\text{dist}(c, L)/\epsilon)$. Every badly-cut client is moved to its closest facility of $L$. Moreover, every client at distance less than $\epsilon \cdot \text{dist}(c, L)$ of its closest facility of $L$ can be moved to it as well. It is then shown that this new instance $I_D$ has small distortion, which essentially means that any solution to $I_D$ can be converted to a solution of the original instance $I$ while only losing a $(1+\epsilon)$-factor in quality. In this instance $I_D$, all clients are cut from their closest facility of $L$ at some level between $\log(\epsilon \cdot \text{dist}(c, L))$ and $\log(\text{dist}(c, L)/\epsilon)$. Using this property, it can be shown that $c$ and its closest center in the optimal solution are also cut at a level in that range. As there are only $O(\log(1/\epsilon))$ levels in this range, by the previous argument, the number of portals is a constant. (See Section 2 for formal definitions and lemmas.)

On highway dimension. The above arguments for doubling metrics hold thanks to Lemma 4. In this work, we show how to construct a similar decomposition for low highway dimension:

▶ **Lemma 5.** Given a shortest-path metric $(V, \text{dist})$ of a graph with highway dimension $h$, a subset $W \subseteq V$, and $\rho > 0$, there exists a polynomial-time computable randomized hierarchical decomposition $D = \{B_0, \ldots, B_{|W|}\} \subseteq W$ such that:

1. **Scaling probability:** for any $v \in V$, radius $r$, and level $i$, we have $\Pr[D \text{ cuts } \beta_v(r) \text{ at level } i] \leq \sigma \cdot r/2^i$, where $\sigma = (h \log(1/\rho))^{O(1)}$.
2. **Interface:** for any $B \in B_i$ on level $i \geq 1$ there exists an interface $I_B \subseteq V$, which is
   a. **concise:** $|I_B| \leq (h/\rho)^{O(1)}$, and
   b. **precise:** for any $u,v \in B$ such that $u$ and $v$ are cut by $D$ at level $i-1$, there exists $p \in I_B$ with $\text{dist}(u,p) + \text{dist}(p,v) \leq \text{dist}(u,v) + 34 \cdot \rho 2^i$.

Our construction relies on the town decomposition from [14], which has the following properties: for a graph with highway dimension $h$ and a given $\rho > 0$, every part $T$ of the decomposition (called a town) has a set $X_T$ of hubs with doubling dimension $O(h \log 1/\rho)$, such that for any two vertices $u$ and $v$ in different child towns of $T$, there is a vertex $x \in X_T$ such that $\text{dist}(u,x) + \text{dist}(x,v) \leq (1 + 2\rho) \cdot \text{dist}(u,v)$ – see Theorem 8 for more details.

This hub set $X_T$ is similar to the portal set of Lemma 4, but has some fundamental differences: the first one is that the decomposition is deterministic, and so it may happen that a client and its facility are cut at a very high level – something that happens only with tiny probability in the doubling setting thanks to the scaling probability. Another main difference is that the size of $X_T$ might be unbounded. As a consequence, it cannot be directly used as a portal set in a dynamic program. To deal with this, we combine the town decomposition with a hierarchical decomposition of each set $X_T$ according to Lemma 4, to build an interface as stated in Lemma 5.

A further notable difference to portals is that the preciseness property of the resulting interface is weaker. In particular, while there is a portal close to each vertex of a part, the hubs can be far from some vertices as long as they lie close to the shortest path to other vertices, which however can be far (due to Lemma 9). As a consequence no analog of near-optimal portal-respecting paths exist. Instead, when connecting a client $c$ with a facility $f$ we need to use the interface point of $I_B$ provided by the preciseness property of Lemma 5 close to the shortest path between $c$ and $f$, where $B$ contains both $c$ and $f$. This shifts the perspective from externally connecting vertices of a part to vertices outside a part, as done for portals, to internally connecting vertices of parts, as done here.
As a consequence, we develop a dynamic program, which follows more or less standard
techniques as for instance given in [3, 18], but needs to handle the weaker preciseness
property of the interface. The main idea is to guess the distances from interface points to
facilities while recursing on the decomposition $D$ of Lemma 5. Due to the shifted perspective
towards internally connecting vertices of parts, the runtime of the dynamic program depends
exponentially on the total number of levels. However, it can be shown that it suffices to
compute a solution on a carefully chosen subset $W$ of the metric for which only a logarithmic
number of levels of the decomposition need to be considered, and thus the runtime is
polynomial.

1.4 Outline

After defining the concepts we use, and stating various structural lemmas in Section 2, we
show how to incorporate our decomposition into the framework of [10]. The proof of Lemma 5
is then presented in Section 3. The formal algorithm is deferred to Section 4. We conclude
the main body of this paper with the hardness proof of Theorem 3 in Section 5.

2 Preliminaries

On doubling metrics. The doubling dimension of a metric is the smallest integer $d$ such
that for any $r > 0$ and $v \in V$, the ball $B(v, 2r)$ of radius $2r$ around $v$ can be covered by at
most $2^d$ balls of half the radius $r$. A doubling metric is a metric space where the doubling
dimension is bounded. In those spaces, one can show the existence of small nets:

► Definition 6. A $\delta$-net of a metric $(V, d)$ is a subset of nodes $N \subseteq V$ with the property
that every node in $V$ is at distance at most $\delta$ from a net point of $N$, and each pair of net
points of $N$ are at distance more than $\delta$.

► Lemma 7 ([16]). Let $(V, d)$ be a metric space with doubling dimension $d$. If its diameter
is $D$, and $N$ is a $\delta$-net of $V$, then $|N| \leq 2^d \left\lfloor \log_2 \left( \frac{D}{\delta} \right) \right\rfloor$. Moreover, any subset $W \subseteq V$ has
doubling dimension at most $2d$.

On highway dimension. We note that for simplicity we will set $c = 8$ in Definition 1
throughout this paper, even if all claimed results are also true for other values of $c$. When we
refer to a metric as having highway dimension $h$, we mean that it is the shortest-path metric
of a graph of highway dimension $h$. The main result we will use about highway dimension is
existence of the following decomposition:

► Theorem 8 ([14]). Given a shortest-path metric $(V, d)$ of highway dimension $h$, and
$p > 0$, there exists a polynomial-time computable deterministic hierarchical decomposition $T$,
called the town decomposition, such that every part $T \in T$, called a town, has a set of hubs\(^3\)
$X_T \subseteq T$ with the following properties:
a. doubling: the doubling dimension of $X_T$ is $d = O(\log(h \log(1/p)))$, and
b. precise: for any two vertices $u$ and $v$ in different child parts of $T$, there is a vertex
$x \in X_T$ such that $d(u, x) + d(x, v) \leq (1 + 2p) \cdot d(u, v)$.

The town decomposition behaves differently from those in Lemmas 4 and 5 in several
ways. The main properties we will need here are the following.

► Lemma 9 ([14]). For any $T \in T$ we have $\text{diam}(T) < \text{dist}(T, V \setminus T)$. Furthermore, for any
child town $T'$ of $T$ we have $\text{diam}(T') \leq \text{diam}(T)/2$.

\(^3\) called approximate core hubs in [14].
On how to incorporate our decomposition into the framework of [10]. Assume we are given an instance $\mathcal{I}$ of $k$-CLUSTERING$^g$ or FACILITY LOCATION$^g$ on some metric $(V, \text{dist})$, together with a hierarchical decomposition $\mathcal{D}$ of the metric with the properties listed in Lemma 5. We start by defining the badly cut clients. In the following, we fix an optimal solution $\text{OPT}$ and an approximate solution $L$, and we define $\tau(\varepsilon, q, \sigma) = \log_2(\sigma(q + 1)^q/\varepsilon^{q+1})$.

Definition 10 (badly cut [10]). Let $(V, \text{dist})$ be a metric of an instance $\mathcal{I}$ of $k$-CLUSTERING$^g$ or FACILITY LOCATION$^g$. $\mathcal{D}$ be a hierarchical decomposition of the metric with scaling probability factor $\sigma$, and $\varepsilon > 0$. If $L_v$ is the distance from $v$ to the closest facility of an approximate solution $L$ to $\mathcal{I}$, then a client $c$ is badly cut w.r.t. $\mathcal{D}$ if the ball $\beta_c(3L_v/\varepsilon)$ is cut as some level $i$ greater than $\log_2(3L_v/\varepsilon) + \tau(\varepsilon, q, \sigma)$.

Similarly, if $\text{OPT}_v$ is the distance from $v$ to the closest facility of the optimum solution $\text{OPT}$ of $\mathcal{I}$, then a facility $f \in L$ is badly cut w.r.t. $\mathcal{D}$ if $\beta_f(3\text{OPT}_f)$ is cut at some level $i$ greater than $\log_2(3\text{OPT}_f) + \tau(\varepsilon, q, \sigma)$.

Given an instance $\mathcal{I}$ of $k$-CLUSTERING$^g$ or FACILITY LOCATION$^g$ and a decomposition $\mathcal{D}$ of the metric, a new instance $\mathcal{I}\mathcal{D}$ is computed to get rid of badly cut clients. The instance $\mathcal{I}\mathcal{D}$ is built from $\mathcal{I}$ by moving clients that are badly cut w.r.t. $\mathcal{D}$ to their closest facility in $L$.

For any client $c$ of $\mathcal{I}\mathcal{D}$ we denote by $\tilde{c}$ the original position of this client in $\mathcal{I}$, i.e., if $\tilde{c}$ is a badly cut client of $\mathcal{I}$ then $c = L(\tilde{c})$ and otherwise $c = \tilde{c}$. The set $F$ of potential centers in unchanged, and thus any solution of $\mathcal{I}$ is a solution of $\mathcal{I}\mathcal{D}$, and vice versa. Note that $\mathcal{I}\mathcal{D}$ does not contain any badly cut client w.r.t. $\mathcal{D}$, and that the definition of $\mathcal{I}\mathcal{D}$ depends on the randomness of $\mathcal{D}$.

To describe the properties we obtain for the new instance, given a solution $S$ to any instance $\mathcal{I}_0$ of $k$-CLUSTERING$^g$ or FACILITY LOCATION$^g$, we define $\text{cost}_{\mathcal{I}_0}(S) = \sum_{v \in V} \chi_{\mathcal{I}_0}(v)\cdot\text{dist}(v, S)$ to be the cost incurred by only the distances to the facilities. Given some $\varepsilon > 0$ and the computed instance $\mathcal{I}\mathcal{D}$ from $\mathcal{I}$, we define

$$\nu_{\mathcal{I}\mathcal{D}} = \max_{\text{solution } S} \left\{ \text{cost}_{\mathcal{I}}(S) - (1 + 2\varepsilon)\text{cost}_{\mathcal{I}\mathcal{D}}(S), (1 - 2\varepsilon)\text{cost}_{\mathcal{I}\mathcal{D}}(S) - \text{cost}_{\mathcal{I}}(S) \right\}.$$ 

If $B_{\mathcal{I}\mathcal{D}}$ denotes the set of badly cut facilities (w.r.t $\mathcal{D}$) of the solution $L$ to $\mathcal{I}$ from which instance $\mathcal{I}\mathcal{D}$ is constructed, we say that $\mathcal{I}\mathcal{D}$ has small distortion w.r.t. $\mathcal{I}$ if $\nu_{\mathcal{I}\mathcal{D}} \leq \varepsilon \text{cost}_{\mathcal{I}}(L)$, and there exists a witness solution $\hat{S} \subseteq F$ that contains $B_{\mathcal{I}\mathcal{D}}$ and for which $\text{cost}_{\mathcal{I}\mathcal{D}}(\hat{S}) \leq (1 + O(\varepsilon))\text{cost}_{\mathcal{I}}(\text{OPT}) + O(\varepsilon)\text{cost}_{\mathcal{I}}(L)$. Moreover, in the case of FACILITY LOCATION$^g$, $\hat{S} = \text{OPT} \cup B_{\mathcal{I}\mathcal{D}}$ and $\sum_{f \in B_{\mathcal{I}\mathcal{D}}} w_f \leq \varepsilon \cdot \sum_{f \in L} w_f$.

Based on these definitions, we now state the main tool we use from [10], and which exploits the scaling probability of our decomposition in Lemma 5 to obtain the required structure.

Lemma 11 ([10]). Let $(V, \text{dist})$ be a metric, and $\mathcal{D}$ be a randomized hierarchical decomposition of $(V, \text{dist})$ with scaling probability factor $\sigma$. Let $\mathcal{I}$ be an instance of $k$-CLUSTERING$^g$ or FACILITY LOCATION$^g$ on $(V, \text{dist})$, with optimum solution $\text{OPT}$ and approximate solution $L$. For any (sufficiently small) $\varepsilon > 0$, with probability at least $1 - \varepsilon$ (over $\mathcal{D}$), the instance $\mathcal{I}\mathcal{D}$ constructed from $\mathcal{I}$ and $L$ as described above has small distortion with a witness solution $\hat{S}$. Furthermore, every client $c$ of $\mathcal{I}\mathcal{D}$ is cut by $\mathcal{D}$ from its closest facility in $\hat{S}$ at level at most $\log_2(3L_v/\varepsilon + 4\text{OPT}_v) + \tau(\varepsilon, q, \sigma)$, where $\tilde{c}$ is the original position of $c$ in $\mathcal{I}$.

As a consequence of Lemma 11, a dynamic program can compute a solution recursively on the parts of $\mathcal{D}$ in polynomial time, as sketched in Section 1.3 and detailed in Section 4.

---

4 More concretely, let $\chi_{\mathcal{I}}$ and $\chi_{\mathcal{I}\mathcal{D}}$ be the demand functions of $\mathcal{I}$ and $\mathcal{I}\mathcal{D}$, respectively. Initially we let $\mathcal{I}\mathcal{D}$ be a copy of $\mathcal{I}$, so that in particular $\chi_{\mathcal{I}\mathcal{D}} = \chi_{\mathcal{I}}$. Then, for each client $c$ of $\mathcal{I}$ that is badly cut in $L$ w.r.t. $\mathcal{D}$, if $L(c)$ denotes the closest facility of $L$ to $c$, in $\mathcal{I}\mathcal{D}$ we set $\chi_{\mathcal{I}\mathcal{D}}(c) = 0$ and increase $\chi_{\mathcal{I}\mathcal{D}}(L(c))$ by the value of $\chi_{\mathcal{I}}(c)$ in $\mathcal{I}$. 

ESA 2020
3 Decomposing the graph

![Figure 1](image-url) A town $T$ and its child towns (black circles). The hubs (crosses) are decomposed by $X_T$ (indicated by different colours). Parts $B \in \mathcal{B}_{i+1}$ (red dashed) are decomposed into parts on level $i$ (pink dashed). Parts of $\mathcal{B}_{i-1}$ can lie in different towns (e.g., the child town of $T$ with subtowns in grey).

This section is dedicated to the proof of Lemma 5. The general idea to construct $\mathcal{D}$ is as follows. For doubling metrics, to decompose a part at level $i$, it is enough to pick a random diameter $\delta \in [2^{-i}, 2^{1-i})$ and divide the part into child parts of diameter $\delta$. This is not doable in the highway dimension setting: if one wishes to decompose a town $T$, it cannot divide any of the child towns, since it is not possible to use the approximate core hubs of $T$ to approximate paths inside one of the child towns. The big picture of our decomposition is therefore as follows. To decompose a town at level $i$, we group randomly (as in the doubling decomposition) the “small” child towns, and put every “big” child town in its own subpart.

As we will see, this turns out to be enough.

In order to decompose a town $T$, we need the following definitions. For each child town $T'$ of $T$ we identify the connecting hub $x \in X_T$, which is some fixed closest hub of $X_T$ to $T'$, breaking ties arbitrarily. Moreover, given a hierarchical decomposition $\mathcal{X}_T = \{\mathcal{U}_0, \ldots, \mathcal{U}_{\lambda(X_T)}\}$ of $X_T$, we define for every $i$ the connecting $i$-cluster of a child town $T'$ of $T$ to be the set $U \in \mathcal{U}_i$ on level $\ell = \min\{i, \lambda(X_T)\}$ containing the connecting hub of $T'$. We then follow the steps below, after choosing $\mu$ from the interval $(0, 1]$ uniformly at random (cf. Figure 1):

1. Given a town $T \in \mathcal{T}$, we apply Lemma 4 to find a randomized hierarchical decomposition $\mathcal{X}_T^* = \{\mathcal{U}_0, \ldots, \mathcal{U}_{\lambda(X_T)}\}$ of the hubs $X_T$ of $T$.

2. Using $\mathcal{X}_T^*$, we define a randomized partial decomposition of $T \cap W$ as follows. For any $i$ and $U \in \mathcal{U}_{\min\{i, \lambda(X_T)\}}$, let the set $\mathcal{A}_i^U \subseteq T \cap W$ be the union of all $T' \cap W$ where $T'$ is a child town of $T$ with the following two properties:
   a. $U$ is the connecting $i$-cluster of $T'$, and
   b. $\text{dist}(T', V \setminus T') \leq \mu 2^i$.

Hence $\mathcal{A}_i^U$ contains all towns somewhat close to $U$, and with small diameter due to Lemma 9. We let $\mathcal{A}_i^U$ be the set containing each non-empty $\mathcal{A}_i^U$.

3. Now, the hierarchical decomposition $\mathcal{D} = \{\mathcal{B}_0, \ldots, \mathcal{B}_{\lambda(W)}\}$ of $W$ can be constructed inductively as follows. At the highest level $\lambda(W)$ of $\mathcal{D}$, $W$ is partitioned in a single set: $\mathcal{B}_{\lambda(W)} = \{W\}$. Now, to decompose a part $B \in \mathcal{B}_{i+1}$ at level $i + 1$, we do the following: Let $T \in \mathcal{T}$ be the inclusion-wise minimal town for which $B \subseteq T$. The “small” subtowns of $T$ lying inside $B$ are grouped according to step (2) (note that $\text{dist}(T', V \setminus T')$ also bounds the diameter of $T'$ by Lemma 9), and the other ones form individual subparts. More formally, the set $\mathcal{B}_i$ contains every part $A \in \mathcal{A}_i^U$ for which $A \subseteq B$, and also every set $T' \cap W$, where $T'$ is a child town of $T$ for which $T' \cap W \subseteq B$ and $T' \cap W$ was not covered by the previously added parts of $\mathcal{A}_i^U$, i.e., $T' \cap W \cap A = \emptyset$ for every $A \in \mathcal{A}_i^U$. 


To prove that the constructed decomposition $\mathcal{D}$ has the desired properties – i.e. that it is indeed a hierarchical decomposition, with parts of bounded diameter and small scaling probability factor –, we begin with some auxiliary lemmas, of which the first one bounds the distance of a town to its connecting hub.

**Lemma 12.** If $T'$ is a child town of $T$ with connecting hub $x \in X_T$, then $\text{dist}(x, T') \leq (1 + 2\rho) \cdot \text{dist}(T', V \setminus T')$.

**Proof.** Let $T''$ be the closest sibling town to $T'$, and let $u \in T'$ and $v \in T''$ be the vertices defining the distance from $T'$ to $T''$, i.e., $\text{dist}(u, v) = \text{dist}(T', T'') = \text{dist}(T', V \setminus T')$. By Theorem 8, there is a hub $y \in X_T$ for which $\text{dist}(u, y) + \text{dist}(y, v) \leq (1 + 2\rho) \cdot \text{dist}(u, v) = (1 + 2\rho) \cdot \text{dist}(T', V \setminus T')$. This implies $\text{dist}(y, T') \leq \text{dist}(u, y) \leq (1 + 2\rho) \cdot \text{dist}(T', V \setminus T')$.

Since the connecting hub $x$ of $T'$ is at least as close to $T'$ as $y$, the claim follows. ▶

Based on the above lemma, we next prove a key property that the diameter of any part of $\mathcal{B}_i \in \mathcal{D}$ is bounded.

**Lemma 13.** If $\rho \leq 1/2$, then the diameter of any part of $\mathcal{B}_i \in \mathcal{D}$ is less than $2^{4+4}$.

**Proof.** On the highest level $\lambda(W)$ of $\mathcal{D}$ the only part of $\mathcal{B}_{\lambda(W)}$ is $W$ itself. As $\lambda(W) = \lfloor \log_2 \text{diam}(W) \rfloor$ we get $\text{diam}(W) \leq 2^{\lambda(W)+1}$, as required.

For any level $i < \lambda(W)$, a set in $\mathcal{B}_i$ is equal to a set $A \in \mathcal{A}_i^0$ for some town $T \in \mathcal{T}$ or it is equal to some set $T' \cap W$ for a child town $T'$ of $T$. In the former case, the set $A$ is equal to a set $A_i^0$ for some cluster $U \in \mathcal{U}_i$ where $i = \min\{i, \lambda(X_T)\}$ and $\mathcal{U}_i \in \mathcal{X}_T$. The set $A_i^0$ contains the union of sets $T' \cap W$ for child towns $T'$ of $T$, for which their connecting hubs lie in $U$ and $\text{dist}(T', V \setminus T') \leq \mu_2^i \leq 2^i$, as $\mu \leq 1$. Thus from Lemma 12 we get $\text{dist}(U, T') \leq (1 + 2\rho)2^i$, and by Lemma 9 we have $\text{diam}(T') < \text{dist}(T', V \setminus T') \leq 2^i$. The cluster $U$ has diameter less than $2^{i+1}$ by Lemma 4, since it is part of the hierarchical decomposition $X_T$ and lies on level $\ell \leq i$. Let $u$ and $v$ be the vertices of $A_i^0$ defining the diameter of $A_i^0$, i.e., $\text{dist}(u, v) = \text{diam}(A_i^0)$. We may reach $v$ from $u$ by first crossing the child town $T'$ that $u$ lies in, then passing over to $U$, then crossing $U$, after which we pass over to the child town $T''$ containing $v$, and finally crossing this child town as well to reach $v$. Hence, assuming that $\rho \leq 1/2$ the diameter of $A_i^0$ is bounded by

$$\text{dist}(u, v) \leq \text{diam}(T') + \text{dist}(U, T') + \text{diam}(U) + \text{dist}(U, T'') + \text{diam}(T'') < 2 \cdot 2^i + (1 + 2\rho)2^i + 2^{i+1} = (6 + 4\rho)2^i \leq 2^{i+3}$$

Now consider the other case, when a set $B \in \mathcal{B}_i$ on level $i < \lambda(W)$ is equal to some set $T' \cap W$ for a child town $T'$ of a town $T$. For such a child town $T'$ there is no enforced upper bound on the distance to other child towns as before, and thus it is necessary to be more careful to bound the diameter of the part. Starting with $B = \mathcal{B}_i$, let $B_i \subseteq B_{i+1} \subseteq \ldots \subseteq B_j$ be the longest chain of parts of increasing levels that are of the same type as $B$. More concretely, for every $\ell \in \{i, i+1, \ldots, j\}$ we have $B_{\ell} \in \mathcal{B}_\ell$ and $B_\ell$ is equal to some set $T_{\ell}' \cap W$ for a child town $T_{\ell}'$ of the inclusion-wise minimal town $T_\ell$ containing $B_{\ell+1}$. Note that in particular $j < \lambda(W)$. As we chose the longest such chain, on the next level $\ell + 1$ there is no such set containing $B_j$, which means that the set $B_{\ell+1} \in \mathcal{B}_{\ell+1}$ for which $B_j \subseteq B_{\ell+1}$ is either equal to a set $A \in \mathcal{A}_{\ell+1}^0$ for some town $T_{\ell+1}'$ containing $B_{\ell+1}$. In either case, from above we get $\text{diam}(B_{\ell+1}) \leq 2^{\ell+4}$.

Note that for any $\ell \in \{i, i+1, \ldots, j-1\}$ the town $T_{\ell}'$ is a descendant town of $T_{\ell+1}'$, since $B_{\ell+1}$ is contained in $T_{\ell+1}'$ and $T_{\ell}'$ is a child town of the inclusion-wise minimal town $T_\ell$ containing $B_{\ell+1}$. By Theorem 8 and Lemma 9 we thus get $\text{diam}(T_{\ell}') \leq \text{diam}(T_{\ell+1}')/2$,
which implies $\text{diam}(T'_i) \leq \text{diam}(T'_j)/2^{j-i}$. The set $B = B_i$ is contained in $T'_i$, which means $\text{diam}(B) \leq \text{diam}(T'_i)$. The town $T'_i$ is the inclusion-wise minimal town containing $B_{j+1}$, while at the same time the child town $T'_j$ of $T'_i$ contains $B_j$. As $B_j \subseteq B_{j+1}$, this means that $B_{j+1}$ both contains vertices inside and outside of $T'_j$, and so $\text{dist}(T'_j, V \setminus T'_j) \leq \text{diam}(B_{j+1})$. By Lemma 9 we know that $\text{diam}(T'_j) \leq \text{dist}(T'_j, V \setminus T'_j)$, and putting all these inequalities together we obtain

\[ \text{diam}(B) \leq \text{diam}(T'_i) \leq \text{diam}(T'_j)/2^{j-i} \leq \text{dist}(T'_j, V \setminus T'_j)/2^{j-i} \leq \text{diam}(B_{j+1})/2^{j-i} \leq 2^{j-i+4}/2^{j-i} = 2^{i+4}. \]

Using Lemma 13 it is not hard to prove the correctness of $\mathcal{D}$, which we turn to next. All statements marked with “*” are deferred to the full version of the paper, due to space constraints.

**Lemma 14 (⋆).** The tuple $\mathcal{D} = \{B_0, \ldots, B_{\lambda(V)}\}$ is a hierarchical decomposition of $W$.

We now turn to proving the properties of Lemma 5, starting with the scaling probability.

**Lemma 15.** The decomposition $\mathcal{D}$ has scaling probability factor $\sigma = (h \log(1/\rho))^{O(1)}$.

**Proof.** To prove the claim, we need to prove that for any $v \in W$, radius $r$, and level $i$, the probability that $\mathcal{D}$ cuts the ball $B_v(r)$ at level $i$ is at most $(h \log(1/\rho))^{O(1)} \cdot r/2^i$. If $\mathcal{D}$ cuts $B_v(r)$ at level $i$, it means that $\beta_v(r)$ is fully contained in a part at level $i+1$: let $T \in \mathcal{T}$ be the inclusion-wise minimal town containing that part. There are two cases to consider: either $\beta_v(r)$ is cut by “small” parts, i.e., there exist two distinct parts $A, A' \in \mathcal{A}_i^T$ such that $v \in A$ and $u \in A'$ for some $u \in W \cap \beta_v(r)$, or not.

We start with the latter case, when $\beta_v(r)$ is not cut by small parts. If $\mathcal{D}$ cuts the ball at level $i$, there are distinct parts $B, B' \in \mathcal{B}_i$ such that $v \in B$ and $u \in B'$ for some $u \in W \cap \beta_v(r)$. Assume w.l.o.g. that $B \notin \mathcal{A}_i^T$ (which is possible to assume since $\beta_v(r)$ is not cut by small parts). By construction of the decomposition, there must be a child town $T' \in \mathcal{T}$, for which $B = T' \cap W$ and $\text{dist}(T', V \setminus T') > \mu/2^i$. Note that $r \geq \text{dist}(v, u) \geq \text{dist}(T', B' \cap T) \geq \text{dist}(T', V \setminus T') > \mu/2^i$, and hence $\mu \leq r/2^i$. The decomposition $\mathcal{D}$ can therefore only cut $\beta_v(r)$ on level $i$ if $\mu < r/2^i$. Since $\mu$ is chosen uniformly at random from the interval $(0, 1]$, the probability is less than $r/2^i$.

We now turn to the other case, when $\beta_v(r)$ is cut by two small parts $A_1$ and $A_2$. The town $T$ must have two child towns $T_1$ and $T_2$ for which $v \in T_1 \cap W \subseteq A_1$ and $u \in T_2 \cap W \subseteq A_2$. Let $x_1$ and $x_2$ be the connecting hubs of $T_1$ and $T_2$. The decomposition $\mathcal{D}$ cuts $v$ and $u$ on level $i$ if and only if $X_T$ cuts $x_1$ and $x_2$ on level $\ell = \min\{i, \lambda(X_T)\}$. Indeed, let $U_1$ and $U_2$ be the connecting $i$-clusters of $T_1$ and $T_2$: then $A_1 = A_{i,1}$ and $A_2 = A_{i,2}$, with $x_1 \in U_1, x_2 \in U_2$. Thus $\mathcal{D}$ cuts $v$ and $u$ on level $i$ if and only if $U_1 \neq U_2$, i.e., if and only if $X_T$ cuts $x_1$ and $x_2$ on level $\ell = \min\{i, \lambda(X_T)\}$.

To compute the probability that $x_1$ and $x_2$ are cut, it is necessary to bound the distance between them. As $v \in T_1$ and $u \in T_2$ while $u \in \beta_v(r)$, for each $j \in \{1, 2\}$ we have $\text{dist}(T_j, V \setminus T_j) \leq \text{dist}(T_1, T_2) \leq r$. By Lemma 12 the distance between $T_j$ and its connecting hub $x_j \in X_T$ is thus at most $(1 + 2\rho)r$. Also, by Lemma 9 we have $\text{diam}(T_j) < \text{dist}(T_j, V \setminus T_j)$, and we get

\[ \text{dist}(x_1, x_2) \leq \text{dist}(x_1, T_1) + \text{diam}(T_1) + \text{dist}(T_1, T_2) + \text{diam}(T_2) + \text{dist}(T_2, x_2) \leq (5 + 4\rho)r. \]

We can reformulate the above as follows: if $\mathcal{D}$ cuts the ball $\beta_v(r)$ at level $i$, and $\beta_v(r)$ is cut by some “small” parts $A_1$ and $A_2$, then $X_T$ cuts the ball $\beta_{x_i}(5 + 4\rho)r$ on level $i$, where $x_1$ is the hub defined for $v$ above. We know that the probability of the latter event
is at most $2^{O(d)}(5 + 4\rho)r/2^i$ by Lemma 4, where $d = O(\log(h \log(1/\rho)))$ is the doubling dimension of $X_T$ by Theorem 8. Hence the probability that $D$ cuts the ball $\beta_i(r)$ at level $i$ is bounded by $(h \log(1/\rho))^{O(1)} \cdot r/2^i$. Taking a union bound over the two considered cases proves the claim.

To prove the remaining property of Lemma 5 for $D$, for each $B \in \mathcal{B}_i$ we need to choose an interface $I_B$ from the whole vertex set $V$. For this we use a carefully chosen net (see Definition 6) of the hubs of the inclusion-wise minimal town $T$ containing $B$, as formalized in the following lemma.

Lemma 16. Given $B \in \mathcal{B}_i$ for some $\mathcal{B}_i \in \mathcal{D}$ and $i \geq 1$, let $T \in \mathcal{T}$ be the inclusion-wise minimal town containing $B$. We define the interface $I_B$ to be a $2\rho^2$-net of the set $Y_B = \{ x \in X_T \mid \text{dist}(x, B) \leq (1 + 2\rho) \text{diam}(B) \}$. The interface $I_B$ has the conciseness and preciseness properties of Lemma 5 for $\rho \leq 1/2$.

Proof. We first prove that $I_B$ is precise. Consider two vertices $u, v \in B$ that are cut at level $i - 1$ by $D$. This means there are two distinct parts $B', B'' \in \mathcal{B}_{i-1}$ on this level such that $v \in B'$ and $u \in B''$. By definition, both $B'$ and $B''$ are unions of sets $T' \cap W$ where $T'$ is a child town of the inclusion-wise minimal town $T$ containing $B$. Also $B' \cap B'' = \emptyset$ by Lemma 14. This means that $T$ has two child towns $T_1$ and $T_2$ for which $v \in T_1 \cap W \subseteq B'$ and $u \in T_2 \cap W \subseteq B''$. By Theorem 8, there is an approximate core hub $x \in X_T$ such that $\text{dist}(u, x) + \text{dist}(x, v) \leq (1 + 2\rho) \text{dist}(u, v)$. In particular, $\text{dist}(x, B) \leq \text{dist}(u, x) \leq (1 + 2\rho) \text{dist}(u, v) \leq (1 + 2\rho) \text{diam}(B)$, as $u, v \in B$. This means that $x \in Y_B$. Since $I_B$ is a $2\rho^2$-net of $Y_B$, there is a hub $p \in I_B$ for which $\text{dist}(x, p) \leq 2\rho^2$. By Lemma 13 we have $\text{dist}(u, v) \leq \text{diam}(B) \leq 2^{i+4} \rho$ if $\rho \leq 1/2$, and so $I_B$ is precise:

$$\text{dist}(u, p) + \text{dist}(p, v) \leq \text{dist}(u, x) + 2 \cdot \text{dist}(x, p) + \text{dist}(x, v) \leq (1 + 2\rho) \text{dist}(u, v) + \rho 2^{i+1} \leq \text{dist}(u, v) + \rho 2 \cdot 2^{i+4} + \rho 2^{i+1} \leq \text{dist}(u, v) + 34 \cdot \rho 2^i.$$

To prove conciseness, recall that $\text{diam}(B) \leq 2^{i+4}$ by Lemma 13, which means that $\text{diam}(Y_B) \leq \text{diam}(B) + 2(1 + 2\rho) \text{diam}(B) \leq 5 \cdot 2^{i+4}$ for $\rho \leq 1/2$. Since $I_B$ is a $2\rho^2$-net of $Y_B$, Lemma 7 implies $|I_B| \leq 2^{d \cdot \log_2(80)/\rho}$, where $d$ is the doubling dimension of $Y_B$. Theorem 8 says that $X_T$ has doubling dimension $O(\log(h \log(1/\rho)))$, and as $Y_B \subseteq X_T$ the same asymptotic bound holds for the doubling dimension $d$ of $Y_B$ by Lemma 7. Therefore we get $|I_B| \leq (h \log(1/\rho))^{O(\log(1/\rho))} = (h/\rho)^{O(1)}$, which concludes the proof.

4. The algorithm

Let an instance $\mathcal{I}$ of the $k$-Clustering or Facility Location problem on a shortest-path metric $(V, \text{dist})$ of a graph $G$ with highway dimension $h$, and maximum demand $X = \max_{v \in V} \chi_\mathcal{I}(v)$ be given. The algorithm performs the following steps:

1. compute a town decomposition $\mathcal{T}$ together with the hubs for each town as given by Theorem 8.
2. compute a hierarchical decomposition $D$ according to Lemma 5, while simultaneously converting $\mathcal{I}$ into a coarse instance w.r.t. $D$, meaning that there is a subset $W \subseteq V$ for which
   a. the clients and facilities of $\mathcal{I}$ are contained in $W$, i.e., $F \cup \{ v \in V \mid \chi_\mathcal{I}(v) > 0 \} \subseteq W$, and
   b. every part of $D$ on level at most $\xi(W) = \lfloor \lambda(W) - 2 \log_2(nX/\varepsilon) \rfloor$ has at most one facility, i.e., $|B \cap F| \leq 1$ for every $B \in \mathcal{B}_\xi(W)$.
3. compute the instance $I_D$ of small distortion as given by Lemma 11.
4. run a dynamic program on $I_D$ as given in Section 4.2, to compute an optimum rounded interface-respecting solution (see Section 4.1 for a formal definition), and output it as a solution to the input instance.

In a nutshell, the coarseness of the instances guarantees that only a logarithmic number of levels need to be considered by the dynamic program. This step loses a $(1+\varepsilon)$-factor in the solution quality. The dynamic program is only able to compute highly structured solutions, which are captured by the notion of rounding and interface-respecting. Due to this, another $(1+\varepsilon)$-factor in the solution quality is lost. In Section 4.1 we prove that the output of the dynamic program is a near-optimal solution to the input instance (proving Theorem 2), and we also detail step (4) of the algorithm. Then in Section 4.2 we describe the details of the dynamic program.

### 4.1 Approximating the distances

One caveat of the dynamic program is that the runtime is only polynomial if the the recursion depth is logarithmic. However when computing our decomposition on the whole metric $(V, \text{dist})$, the number of levels is $\lambda(V) + 1 = \lceil \log_2 \text{diam}(V) \rceil + 1$, which can be linear in the input size. Standard techniques can be used to reduce the number of levels to $O(\log(n/\varepsilon))$ when aiming for a $(1+\varepsilon)$-approximation by preprocessing the input metric. However, for graphs of bounded highway dimension these general techniques change the hub sets and we would have to be careful to maintain the properties we need, as given by Theorem 8. Therefore we adapt the standard techniques to our setting via the notion of coarse instances.

The following lemma shows that we can reduce any instance to a set of coarse ones, for which, as we will see, our dynamic program only needs to consider the highest $2\log_2(nX/\varepsilon)$ levels.

**Lemma 17.** Let $I$ be an instance of $k$-CLUSTERING$^a$ or FACILITY LOCATION$^a$ on a graph $G$ of highway dimension $h$. There are polynomial-time computable instances $I_1, \ldots, I_b$ and respective hierarchical decompositions $D_1, \ldots, D_b$ with the properties given in Lemma 5 for any $\rho \leq 1/2$, such that for each $i \in \{1, \ldots, b\}$ the instance $I_i$ is also defined on $G$ and is coarse w.r.t. $D_i$. Furthermore, if an $\alpha$-approximation can be computed for each of the instances $I_1, \ldots, I_b$ in polynomial time, then for any $\varepsilon > 0$ a $(1+O(\varepsilon))\alpha$-approximation can be computed for $I$ in polynomial time.

Lemma 17 implies that if there is a PTAS for coarse instances, we also have a PTAS in general. Hence from now on we assume that the given instance $I$ is coarse w.r.t. a hierarchical decomposition $D$ of some subset $W$ of the vertices of the input graph $G$, where $D$ has bounded scaling probability factor and concise and precise interface sets in $G$ according to Lemma 5 (for some value $\rho > 0$ specified later).

The next step of the algorithm is to compute a new instance $I_D$ with small distortion as given by Lemma 11. Recall that $I_D$ is obtained from $I$ by moving badly cut clients to facilities of $L$. In particular, the instance $I_D$ is also coarse w.r.t. $D$, which means that we may run our dynamic program on $I_D$.

The dynamic program exploits the interface sets of $D$ by computing a near-optimum “interface-respecting” solution to $I_D$, i.e., a solution where clients are connected to facilities through interface points. Moreover, for the dynamic program to run in polynomial time it can only estimate the distances between interface points and facilities to a certain precision. In general, we denote by $\langle x \rangle_i = \min\{(35+\delta)\rho 2^i \mid \delta \in \mathbb{N} \text{ and } \rho \delta 2^i \geq x\}$ the value of $x$ rounded to
the next multiple of $\rho 2^i$ and shifted by $35\rho 2^i$. We then define the rounded interface-respecting distance $\text{dist}'(v, u)$ from a vertex $v$ to another vertex $u$ as follows. If $v$ and $u$ are not cut at any level, i.e., $v = u$, then $\text{dist}'(v, u) = 0$. Otherwise, if $i \geq 1$ is the level of $D$ such that there is a part $B \in B_i$ with $v, u \in B$, and $D$ cuts $v$ and $u$ at level $i - 1$, we let

$$\text{dist}'(v, u) = \min \{ \text{dist}(v, p) + \langle \text{dist}(p, u) \rangle, \mid p \in I_B \}.$$  

Note that $\text{dist}'(\cdot, \cdot)$ does not necessarily fulfill the triangle inequality, and is also not symmetric. We therefore need the bounds of the following lemma.

**Lemma 18.** For any level $i \geq 1$ and vertices $v$ and $u$ that are cut by $D$ on level $i - 1$ we have $\text{dist}'(v, u) \leq \text{dist}(v, u) + 70 \cdot \rho 2^i$. Let $B \in B_j$ be the part on some level $j \geq i$ with $v, u \in B$. For any $p \in I_B$ we have $\text{dist}'(v, u) \leq \text{dist}(v, u) + \langle \text{dist}(p, u) \rangle_j$.

**Proof.** Let $B' \in B_i$ be the part on level $i$ containing both $v$ and $u$. By Lemma 5 there is an interface point $p' \in I_{B'}$ such that $\text{dist}(v, p') + \text{dist}(p', u) \leq \text{dist}(v, u) + 34 \cdot \rho 2^i$. By definition of the rounding we also have $\langle \text{dist}(p', u) \rangle_i \leq \text{dist}(p', u) + 36 \cdot \rho 2^i$. Hence $\text{dist}'(v, u) \leq \text{dist}(v, p') + \langle \text{dist}(p', u) \rangle_i \leq \text{dist}(v, p') + \text{dist}(p', u) + 36 \cdot \rho 2^i \leq \text{dist}(v, u) + 70 \cdot \rho 2^i$.

The second part is obvious if $j = i$ from the definition of $\text{dist}'(v, u)$. If $j > i + 1$, we use the above bound on $\text{dist}'(v, u)$ together with the additive shift of the rounding and the triangle inequality of $\text{dist}(\cdot, \cdot)$ to obtain

$$\text{dist}'(v, u) \leq \text{dist}(v, u) + 70 \cdot \rho 2^i \leq \text{dist}(v, p) + \text{dist}(p, u) + 70 \cdot \rho 2^{i-1}$$

$$\leq \text{dist}(v, p) + \langle \text{dist}(p, u) \rangle_j - 35 \cdot \rho 2^j + 70 \cdot \rho 2^{j-1} = \text{dist}(v, p) + \langle \text{dist}(p, u) \rangle_j.$$

For any non-empty set $S$ of facilities, we define $\text{dist}'(v, S) = \min_{f \in S} \{ \text{dist}'(v, S) \}$, and for empty sets we let $\text{dist}'(v, \emptyset) = \infty$. Analogous to $\text{cost}_{\mathcal{I}_D}(S)$, for a solution $S$ to some instance $\mathcal{I}_0$ we also define $\text{cost}'_{\mathcal{I}_D}(S)$ using $\text{dist}'(\cdot, \cdot)$ as

$$\text{cost}'_{\mathcal{I}_D}(S) = \sum_{v \in V} \chi_{\mathcal{I}_D}(v) \cdot \text{dist}'(v, S)^q.$$  

We show the following lemma, which translates between $\text{cost}'_{\mathcal{I}_D}$ and $\text{cost}_{\mathcal{I}_D}$, and is implied by the preciseness of the interface sets and the fact that $\mathcal{I}_D$ has small distortion. Recall that the set of facilities is the same in $\mathcal{I}$ and $\mathcal{I}_D$, i.e., a solution to one of these instances is also a solution to the other.

**Lemma 19 (•).** Let $\mathcal{I}$ be an instance of $k$-CLUSTERING or FACILITY LOCATION with optimum solution $OPT$ and approximate solution $L$. Let $\mathcal{I}_D$ be an instance of small distortion for some $0 < \varepsilon < 1/2$, computed from $L$ and a hierarchical decomposition $D$ with precise interface sets for $\rho \leq \frac{\varepsilon^{\alpha+1}}{200(q+1)^2}$ according to Lemma 5. For the witness solution $\hat{S}$ of $\mathcal{I}_D$ we have $\text{cost}'_{\mathcal{I}_D}(\hat{S}) \leq (1 + O(\varepsilon)) \text{cost}_{\mathcal{I}_D}(OPT) + O(\varepsilon) \text{cost}_{\mathcal{I}_D}(L)$. Moreover, for any solution $S$ we have $\text{cost}_{\mathcal{I}_D}(S) \leq (1 + O(\varepsilon)) \text{cost}'_{\mathcal{I}_D}(S) + O(\varepsilon) \text{cost}_{\mathcal{I}_D}(L)$.

The next lemma states the properties of the dynamic program that for any coarse instance $\mathcal{I}_0$ computes an optimal rounded interface-respecting solution, which formally is a subset $OPT'$ of facilities that minimizes $\text{cost}'_{\mathcal{I}_0}(OPT')$ with $|OPT'| \leq k$ for $k$-CLUSTERING, while for FACILITY LOCATION it minimizes $\text{cost}'_{\mathcal{I}_0}(OPT') + \sum_{f \in OPT'} w_f$. This step of the algorithm exploits the conciseness of the interface sets and the coarseness of the instance to bound the runtime. We prove the following lemma in Section 4.2.
Lemma 20. Let $I_0$ be an instance of $k$-Clustering or Facility Location that for some $\varepsilon > 0$ is coarse w.r.t. a hierarchical decomposition $\mathcal{D}$ with concise interface sets for some $1/2 \geq \rho > 0$ according to Lemma 5. An optimum rounded interface-respecting solution for $I_0$ can be computed in $(nX/\varepsilon)^{(h/\rho)O(1)}$ time.

We are now ready to put together the above lemmas to prove Theorem 2. Due to space constraints however, the formal proof is deferred to the full version of the paper.

4.2 The dynamic program (proof of Lemma 20)

We describe the algorithm for $k$-Clustering, and only mention in the end how to modify the algorithm to compute a solution for Facility Location.

The solution is computed by a dynamic program recursing on the decomposition $\mathcal{D}$. Let $W$ be the vertex set that $\mathcal{D}$ decomposes, and which contains all clients and facilities of the coarse instance $I$. Roughly speaking, the table of the dynamic program will have an entry for every part $B \in B_i$ of $\mathcal{D}$ on all levels $i \geq \xi(W)$, for which it will estimate the distance from each interface point on all higher levels $j \geq i + 1$ to the closest facility of the optimum solution. That is, if $\tilde{B} \in B_j$ is a higher-level part for which $B \subseteq \tilde{B}$, then the distances from all interface points $I_{\tilde{B}}$ to facilities of the solution in $\tilde{B}$ will be estimated.

Here the estimation happens in two ways. First off, the distances to facilities outside of $B$ have to be guessed. That is, there is an external distance function $d_j^+$ that assigns a distance to each interface point of $I_{\tilde{B}}$, anticipating the distance from such a point to the closest facility of $\tilde{B}$, if this facility lies outside of $B$. In order to verify whether the guess was correct, each entry for a part $B$ on level $i$ also provides an internal distance function $d_j^-$, which stores the distance from each interface point of $I_{\tilde{B}}$ on level $j \geq i + 1$ to the closest facility, if the facility is guessed to lie inside of $B$.

The other way in which distances are estimated concerns the preciseness with which they are stored. The distance functions $d_j^+$ and $d_j^-$ will only take rounded values $(\bar{x})_j$, where $0 < x \leq 2^{j+5}$, or $\infty$ if no facility at the appropriate distance exists. In particular, if the facility of the solution in $\tilde{B}$ that is closest to $p \in I_{\tilde{B}}$ lies outside of $B$ then $d_j^-(p) = \infty$. If there is no facility of the solution in $\tilde{B}$ then both distance functions $d_j^+$ and $d_j^-$ are set to $\infty$ for all $p \in I_{\tilde{B}}$. Note that this means that at least one of $d_j^+(p)$ and $d_j^-(p)$ is always set to $\infty$. Note also that the finite values in the domains of the distance functions admit to store the rounded distance to any facility in $\tilde{B}$ on level $j$, since the diameter of $\tilde{B}$ is at most $2^{j+4}$ by Lemma 13, and the distance from any $p \in I_{\tilde{B}}$ to $\tilde{B}$ is at most $(1 + 2\rho)\text{diam}(\tilde{B})$ by Lemma 16, i.e., for any $f \in \tilde{B} \cap F$ we have $\text{dist}(p, f) \leq (1 + 2\rho)2^{j+4} \leq 2^{j+5}$ using $\rho \leq 1/2$.

Formal definition of the table. Let us denote by $I_B$ the interface set of the part $B \in B_j$ on level $j \geq i + 1$ containing $B \in B_i$, i.e., $I_B = I_{\tilde{B}}$. Every entry of the dynamic programming table $T$ is defined by a part $B \in B_i$ of $\mathcal{D}$ on a level $i \in \{\xi(W), \ldots, \lambda(W)\}$, and two distance functions $d_j^+, d_j^- : I_B \rightarrow \{\langle x \rangle_j \mid 0 < x \leq 2^{j+5}\} \cup \{\infty\}$ for each $j \in \{i + 1, \ldots, \lambda(W)\}$, such that max\{$d_j^+(p), d_j^-(p)$\} = $\infty$ for all $p \in I_B$. Additionally, each entry comes with an integer $k' \in \{0, \ldots, k\}$, which is a guess on the number of facilities that the optimum solution contains in $B$.

In an entry $T[B, k', (d_j^+, d_j^-)^{\lambda(W)}_{j=i+1}]$ we store the rounded interface-respecting cost of connecting the clients of $B$ to facilities that adhere to the distance functions. More concretely, let $S \subseteq F \cap B$ be any subset of facilities in $B$. We say that $S$ is compatible with an entry $T[B, k', (d_j^+, d_j^-)^{\lambda(W)}_{j=i+1}]$ if $|S| = k'$, and for any $j \geq i + 1$ the values of the distance functions for every interface point $p \in I_B$ are set to either

- $d_j^-(p) = \langle \text{dist}(p, S) \rangle_j$ and $d_j^+(p) = \infty$, or
- $d_j^+(p) \leq \langle \text{dist}(p, S) \rangle_j$ and $d_j^-(p) = \infty$. 

Recall that \( \text{dist}(v, \emptyset) = \infty \), and so the empty set \( S = \emptyset \) is compatible with an entry \( T[B, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) if \( k' = 0 \), and the values of all internal distance functions are set to \( \infty \).

Over all sets \( S \subseteq F \cap B \) compatible with the entry \( T[B, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) for \( B \in B_i \), the entry should store the minimum value of \( C_B(S) \), which is defined as

\[
C_B(S) = \sum_{v \in B} \chi_{\Delta_0}(v) \cdot \min \left\{ \text{dist}'(v, S), \min_{p \in I_B} \left\{ \text{dist}(v, p) + d_j^+(p) \right\} \right\}.
\]

If there is no compatible set \( S \subseteq F \cap B \) for the entry, then \( T[B, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) is equal to \( \infty \).

On the highest level \( i = \lambda(W) \), there are no distance functions to adhere to on levels \( j \geq i + 1 \), and thus any set \( S \subseteq W \) of facilities is compatible with the entry for \( B = W \) and \( k' = |S| \). Furthermore, \( \text{cost}'_{\Delta_0}(S) \) is equal to \( C_W(S) \), since \( W \) contains all clients and facilities of the coarse instance \( \Delta_0 \). In particular, the entry of \( T \) for which \( k' = k \) and \( B = W \), will contain the objective function value of the optimum rounded interface-respecting solution to \( \Delta_0 \). Hence if we can compute the table \( T \) we can also output the optimum rounded interface-respecting solution via this entry.

**Computing the table.** We begin with a part \( B \in B_{\xi(W)} \) on the lowest considered level \( \xi(W) \), for which we know that \( B \) contains at most one facility, as \( \Delta_0 \) is coarse. If \( B \) contains no facility, then only \( S = \emptyset \) can be compatible with the entry \( T[B, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) and computing the value of the entry is straightforward given the definition of \( C_B(S) \), where all incompatible entries are set to \( \infty \). If \( B \) contains one facility \( f \), then any compatible set \( S \) is either empty or only contains \( f \). We can thus check whether either of the two options is compatible with the entry \( T[B, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) by checking if \( k' \) is set to 0 or 1, respectively, and checking that all values of the internal distance function are set correctly. Thereafter we can again use the definition of \( C_B(S) \) to compute the values for both possible sets \( S \) and store them in the respective compatible entries. All incompatible entries are set to \( \infty \).

Now fix a part \( B \in B_i \) that lies on a level \( i > \xi(W) \). We show how to compute all entries \( T[B, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) for all values \( k' \) and distance functions. By induction we have already computed the correct values of all entries of \( T \) for parts \( B' \in B_{i-1} \) where \( B' \subseteq B \). We order these parts arbitrarily, so that \( B'_1, \ldots, B'_l \) are the parts of \( B_{i-1} \) contained in \( B \). We then define an auxiliary table \( \hat{T} \) that is similar to the table \( T \), but should compute the best compatible facility set in the union \( B'_{\leq \ell} = \bigcup_{h=1}^{\ell} B'_h \) of the first \( \ell \) subparts of \( B \). Accordingly, \( \hat{T} \) has an entry for each union of parts \( B'_{\leq \ell} \), each \( k' \in \{0, \ldots, k\} \), and distance functions \( d_{j,1}^+, d_{j,1}^- : I_B \rightarrow \{0, 1\} \) for each \( j \in \{0, \ldots, \lambda(W)\} \), such that \( \max\{d_{j,1}^+(p), d_{j,1}^-(p)\} = \infty \) for all \( p \in I_B \). Here, naturally, \( I_B = I_B \), i.e., the entry also takes the interface set of \( B \) into account.

Analogous to before, a set \( S \subseteq F \cap B'_{\leq \ell} \) of facilities in the union is compatible with an entry \( \hat{T}[B'_{\leq \ell}, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) if \( |S| = k' \), and for any \( j \geq i \) the values of the distance functions for every interface point \( p \in I_B \) are set to either

\[
\text{dist}(p, S) = \infty \quad \text{or} \quad d_{j,1}^-(p) = \infty
\]

or

\[
\text{dist}(p, S) = \infty \quad \text{or} \quad d_{j,1}^+(p) = \infty
\]

Over all sets \( S \subseteq F \cap B'_{\leq \ell} \) compatible with \( \hat{T}[B'_{\leq \ell}, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \), the entry should store the minimum value of \( \hat{C}_{\leq \ell}(S) \), which is defined as

\[
\hat{C}_{\leq \ell}(S) = \sum_{v \in B'_{\leq \ell}} \chi_{\Delta_0}(v) \cdot \min \left\{ \min_{p \in I_B} \left( \text{dist}(v, p) + d_{j,1}^+(p) \right) \right\}.
\]

If there is no compatible set \( S \subseteq F \cap B'_{\leq \ell} \) for the entry, then \( \hat{T}[B'_{\leq \ell}, k', (d_{j,1}^+, d_{j,1}^-)]_{j=1}^{\lambda(W)} \) is equal to \( \infty \).
To compute $T$ using the auxiliary table $\hat{T}$, note that since $B = B'_{\leq b}$, any set $S \subseteq F \cap B$ is compatible with the entry $T[B, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}]$ if and only if it is compatible with a corresponding entry $\hat{T}[B'_{\leq b}, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}]$ for some internal distance function $d_i^-$ on level $i$. Furthermore, if $d_i^+(p) = \infty$ for all $p \in F$, then $C_B(S) = \hat{C}_B(S)$ for such a set $S$. Therefore we can easily compute the entry $T[B, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}]$ from $\hat{T}$ by setting

$$T[B, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}] = \min_{d_j} \left\{ \hat{T}[B'_{\leq b}, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}] \mid \forall p \in I_B : d_i^+(p) = \infty \right\}.$$  

Computing the auxiliary table. Also computing an entry of $\hat{T}$ for $B'_{\leq 1}$ is easy using the entries of $T$ for $B'_1$, since $B'_1 = B'_{\leq 1}$ and so (taking the index shift of $i$ into account) we have

$$\hat{T}[B'_{\leq 1}, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}] = T[B'_1, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}].$$

To compute entries of $\hat{T}$ for some $B'_{\leq \ell}$ where $\ell \geq 2$, we combine entries of table $T$ for $B'_1$ with entries of table $\hat{T}$ for $B'_\leq \ell-1$. However we will only combine entries with distance functions that imply compatible solutions. More concretely, we say that distance functions $(d_j^+, d_j^-)_{j=i}^{\lambda(W)}$ for $B'_{\leq \ell}$, $(\delta_j^+, \delta_j^-)_{j=i}^{\lambda(W)}$ for $B'_1$, and $(\beta_j^+, \beta_j^-)_{j=i}^{\lambda(W)}$ for $B'_{\leq \ell-1}$ are consistent if for every level $j \geq i$ and $p \in I_B$ we have one of

1. $d_j^+(p) = \delta_j^+(p) = \delta_j^-(p) = \beta_j^+(p) = \beta_j^-(p) = \infty$, or
2. $d_j^-(p) = \delta_j^-(p) = \beta_j^-(p) = \infty$, or
3. $d_j^+(p) = \delta_j^+(p) = \beta_j^+(p) = \infty$, or

The algorithm now considers all sets of consistent distance functions to compute an entry $\hat{T}[B'_{\leq \ell}, k', (d_j^+, d_j^-)_{j=i}^{\lambda(W)}]$ for $\ell \geq 2$ by setting it to

$$\min \left\{ T[B'_1, k'', (\delta_j^+, \delta_j^-)_{j=i}^{\lambda(W)}] + \hat{T}[B'_{\leq \ell-1}, k' - k'', (\beta_j^+, \beta_j^-)_{j=i}^{\lambda(W)}] \mid k'' \in \{0, \ldots, k'\} \text{ and } (d_j^+, d_j^-)_{j=i}^{\lambda(W)}, (\delta_j^+, \delta_j^-)_{j=i}^{\lambda(W)}, (\beta_j^+, \beta_j^-)_{j=i}^{\lambda(W)} \text{ are consistent} \right\} \quad (1)$$

We now prove the correctness using two lemmas. The following lemma implies that if we only consider consistent distance functions to compute entries recursively, then the entries will store values for compatible solutions.

\begin{lemma}
Let $(d_j^+, d_j^-)_{j=i}^{\lambda(W)}$ for $B'_{\leq \ell}$, $(\delta_j^+, \delta_j^-)_{j=i}^{\lambda(W)}$ for $B'_1$, and $(\beta_j^+, \beta_j^-)_{j=i}^{\lambda(W)}$ for $B'_{\leq \ell-1}$ be consistent distance functions, and let $S_1 = B'_1 \cap F$ and $S_2 = B'_{\leq \ell-1} \cap F$ be facility sets. If $S_1$ is compatible with entry $T[B'_1, |S_1|, (\delta_j^+, \delta_j^-)_{j=i}^{\lambda(W)}]$ and $S_2$ is compatible with entry $\hat{T}[B'_{\leq \ell-1}, |S_2|, (\beta_j^+, \beta_j^-)_{j=i}^{\lambda(W)}]$, then the union $S = S_1 \cup S_2$ is compatible with entry $\hat{T}[B'_{\leq \ell}, |S|, (d_j^+, d_j^-)_{j=i}^{\lambda(W)}]$. Moreover, $\hat{C}_\leq(S) = C_{B'_\ell}(S_1) + \hat{C}_{\leq \ell-1}(S_2).$
\end{lemma}

\begin{proof}
To prove compatibility of $S$ with the entry $\hat{T}[B'_{\leq \ell}, |S|, (d_j^+, d_j^-)_{j=i}^{\lambda(W)}]$, it suffices to show that the distance functions are set correctly. Fix a level $j \geq i$ and an interface point $p \in I_B$. There are three cases to consider, according to the definition of consistency of the distance functions. In the first case, all three internal distance functions are set to $\infty$, and all external distance functions are set to the same value. In particular, since $S_1$ and $S_2$ are compatible with their respective entries, we have $d_j^+(p) = \delta_j^+(p) = \beta_j^+(p) \leq \min \{ \langle \text{dist}(p, S_1) \rangle_j, \langle \text{dist}(p, S_2) \rangle_j \} = \langle \text{dist}(p, S) \rangle_j$, as $S = S_1 \cup S_2$. In the second case, $\beta_j^-(p) = \delta_j^+(p) = \infty$ and so $\beta_j^+(p) \leq \langle \text{dist}(p, S_2) \rangle_j$, since $S_2$ is compatible with its entry, and
\( \delta_j^\prime (p) = \langle \text{dist}(p, S_1) \rangle_j, \) since \( S_1 \) is compatible with its entry. Since we also have \( \beta_j^\prime (p) = \delta_j^\prime (p) \) we get \( \langle \text{dist}(p, S_1) \rangle_j \leq \langle \text{dist}(p, S_2) \rangle_j \), and hence \( \langle \text{dist}(p, S) \rangle_j = \langle \text{dist}(p, S_1) \rangle_j \). Consistency furthermore implies \( \delta_j^\prime (p) = \delta_j^\prime (p) = \langle \text{dist}(p, S) \rangle_j \) and \( \delta_j^\prime (p) = \infty \). The third case is analogous to the second, and therefore \( S \) is compatible with its entry.

For the second part, we consider the contributions of vertices to the terms \( \hat{C}_{\leq \ell-1}(S) \), \( C_{B_1^\prime}(S_1) \), and \( \hat{C}_{\leq \ell-1}(S_2) \), and show that they are the same for \( \hat{C}_{\leq \ell}(S) \) and for \( C_{B_1^\prime}(S_1) + \hat{C}_{\leq \ell-1}(S_2) \). For this we first fix a vertex \( v \in B_{\leq \ell-1}' \), and in the following distinguish the cases where its contribution to \( \hat{C}_{\leq \ell-1}(S_2) \) and \( \hat{C}_{\leq \ell}(S) \) is due to a facility or an interface point.

The first case is that \( \text{dist}'(v, S_2) \leq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + \beta_j^\prime (p) \} \), i.e., the contribution of \( v \) to \( \hat{C}_{\leq \ell-1}(S_2) \) is given by a facility of \( S_2 \). Note that the consistency of the distance functions always implies that \( \beta_j^\prime (p) = d_j^\prime (p) \) or \( d_j^\prime (p) = \infty \) for any level \( j \geq i \) and interface point \( p \in I_B \), and so \( \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + \beta_j^\prime (p) \} \leq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + d_j^\prime (p) \} \). At the same time \( \text{dist}'(v, S) \leq \text{dist}'(v, S_2) \) as \( S_2 \subseteq S \). We hence get that \( \text{dist}'(v, S) \leq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + d_j^\prime (p) \} \), i.e., the contribution of \( v \) to \( \hat{C}_{\ell}(S) \) is also given by a facility of \( S \) in this case. Thus to show that the contribution of \( v \) to \( \hat{C}_{\leq \ell-1}(S_2) \) and \( \hat{C}_{\ell}(S) \) is the same, we need to show that \( \text{dist}'(v, S) = \text{dist}'(v, S_2) \). Note that this is implied if \( \text{dist}'(v, S) \geq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + \beta_j^\prime (p) \} \), since we have \( \text{dist}'(v, S) \leq \text{dist}'(v, S_2) \leq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + \beta_j^\prime (p) \} \). Thus the following proves the claim, using that the contribution of \( v \) to \( \hat{C}_{\ell}(S) \) is given by a facility of \( S \).

\[ \triangleright \] Claim 22. For \( v \in B_{\leq \ell-1}' \), if \( \text{dist}'(v, S) \leq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + d_j^\prime (p) \} \) then we have \( \text{dist}'(v, S) = \text{dist}'(v, S_2) \) or \( \text{dist}'(v, S) \geq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + \beta_j^\prime (p) \} \).

**Proof.** Given \( \text{dist}'(v, S) \leq \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + d_j^\prime (p) \} \), assume to the contrary that we have \( \text{dist}'(v, S) < \min_{j \geq i, p \in I_B^\prime} \{ \text{dist}(v, p) + \beta_j^\prime (p) \} \) and \( \text{dist}'(v, S) \neq \text{dist}'(v, S_2) \), which, as \( S = S_1 \cup S_2 \), means \( \text{dist}'(v, S) < \text{dist}'(v, S_2) \). The latter inequality implies that the value of \( \text{dist}'(v, S) \) is obtained for some facility \( f \in S_1 \subseteq B_1' \). In particular, \( v \in B_{\leq \ell-1}' \) and \( f \in B_1' \) are cut at level \( i-1 \), and so there is an interface point \( p \in I_B^\prime \) such that \( \text{dist}'(v, S) = \text{dist}(v, p) + \langle \text{dist}(p, f) \rangle_i \), and \( f \) is the closest facility to \( p \) in \( S \), i.e., \( \langle \text{dist}(p, S) \rangle_i = \langle \text{dist}(p, f) \rangle_i \). Using the former of the assumed inequalities we get \( \text{dist}(v, p) + \langle \text{dist}(p, f) \rangle_i = \text{dist}'(v, S) < \text{dist}(v, p) + \beta_j^\prime (p) \), and so we can conclude that \( \langle \text{dist}(p, f) \rangle_i < \beta_j^\prime (p) \).

Using the inequality of the premise of the claim, we also get \( \text{dist}(v, p) + \langle \text{dist}(p, f) \rangle_i = \text{dist}'(v, S) \leq \text{dist}(v, p) + d_j^\prime (p) \), i.e., \( \langle \text{dist}(p, f) \rangle_i \leq d_j^\prime (p) \). Since \( S \) is compatible with entry \( T[B_{\leq \ell-1}', |S|, (d_j^\prime, d_j^\prime, j=1, \ldots, |W|)] \), we have \( d_j^\prime (p) = \infty \) or \( d_j^\prime (p) \leq \langle \text{dist}(p, S) \rangle_i \). In the latter case we would have \( d_j^\prime (p) \leq \langle \text{dist}(p, S) \rangle_i = \langle \text{dist}(p, f) \rangle_i < \beta_j^\prime (p) \), which however cannot happen if the distance functions are consistent. Thus compatibility of \( S \) implies \( d_j^\prime (p) = \infty \) and \( d_j^\prime (p) = \langle \text{dist}(p, f) \rangle_i \). In particular, we can conclude that \( d_j^\prime (p) \) has a finite value (as \( f \) exists) and \( \beta_j^\prime (p) \) differs from \( d_j^\prime (p) \). This can only mean that the third of the consistency properties applies to \( p \) at level \( i \), and so \( \beta_j^\prime (p) = d_j^\prime (p) = \langle \text{dist}(p, f) \rangle_i \).

In particular, also \( \beta_j^\prime (p) \) has a finite value, and using the compatibility of \( S_2 \) with entry \( T[B_{\leq \ell-1}', |S_2|, (\beta_j^\prime, \beta_j^\prime, j=1, \ldots, |W|)] \), we can conclude that there exists a facility \( f' \in S_2 \subseteq B_{\leq \ell-1}' \) with \( \langle \text{dist}(p, f') \rangle_i = \beta_j^\prime (p) = \langle \text{dist}(p, f) \rangle_i \). Now let \( j \leq i \) be the level for which \( v \in B_{\leq \ell-1}' \) and \( f' \in B_{\leq \ell-1}' \) are cut at level \( j-1 \) by \( D \). Lemma 18 implies \( \text{dist}'(v, f') \leq \text{dist}(v, p) + \langle \text{dist}(p, f') \rangle_i \), but then we have

\[
\text{dist}'(v, S_2) \leq \text{dist}'(v, f') \leq \text{dist}(v, p) + \langle \text{dist}(p, f') \rangle_i = \text{dist}(v, p) + \langle \text{dist}(p, f) \rangle_i = \text{dist}'(v, S),
\]

which is a contradiction to \( \text{dist}'(v, S) < \text{dist}'(v, S_2) \). \( \triangleright \)
The next case we consider is that \( \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + d_j^+\} \leq \text{dist}'(v, S) \), i.e., the contribution of \( v \) to \( \tilde{C}_t(S) \) is given by an interface point. As observed before, we have \( \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \leq \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + d_j^+(p) \} \) and \( \text{dist}'(v, S) \leq \text{dist}(v, S_2) \), which implies \( \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \leq \text{dist}(v, S_2) \), i.e., in this case the contribution of \( v \) to \( \tilde{C}_{t-1}(S_2) \) is also given by an interface point. Note that it also implies \( \text{dist}'(v, S) > \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \), and thus the following claim shows that the contribution of \( v \) to \( \tilde{C}_t(S) \) and \( \tilde{C}_{t-1}(S_2) \) is the same.

\[ \triangleright \text{Claim 23.} \quad \text{For } v \in B'_{t-1}, \text{ if } \text{dist}'(v, S) > \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \text{ then we have } \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} = \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + d_j^+(p) \}. \]

Proof. Given \( \text{dist}'(v, S) > \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \), assume to the contrary that \( \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \neq \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + d_j^+(p) \} \). As observed above, the consistency of the distance functions always implies \( \beta_j^+(p) = d_j^+(p) \) or \( d_j^+(p) = \infty \), and thus we must have \( \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} < \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + d_j^+(p) \} \). Let \( j \geq i \) and \( p \in I_j \) be the level and interface point for which the minimum of the former term of this inequality is obtained. The inequality then implies \( \beta_j^+(p) < d_j^+(p) \) for this particular point \( p \) and level \( j \), which can only be the case if \( \beta_j^+(p) < \infty \) and \( d_j^+(p) = \infty \). The values of \( \beta_j^+(p) \) and \( d_j^+(p) \) can only differ if the second of the consistency properties applies to \( p \) at level \( j \), and so \( \beta_j^+(p) = d_j^+(p) \). Since \( \beta_j^+(p) < \infty \), the compatibility of \( S \) with entry \( \tilde{T}(S, |S|, (d_j^+, d_j^-)\}_{j=1}^{\infty} \) implies \( \beta_j^+(p) = d_j^+(p) = \text{dist}(p, S) \).

Now let \( f \in S \subseteq B'_{t-1} \) be the facility for which \( \text{dist}'(v, S) = \text{dist}'(v, f) \) (which exists as \( d_j^+(p) < \infty \)). Let \( j' \leq j \) be the level for which \( v \in B'_{t-1} \) and \( f \in B'_{j'} \) are cut at level \( j' - 1 \) by \( \tilde{D} \). By Lemma 18 we have \( \text{dist}'(v, f) \leq \text{dist}(v, p) + \langle \text{dist}(p, f) \rangle_j \), since \( j' \leq j \) and the part \( B \in B_j \), containing \( v \) and \( f \) is itself contained in some part \( B' \in B_j \), with \( v, f \in B' \) and \( p \in I_j \). But then,

\[ \text{dist}'(v, S) \leq \text{dist}(v, p) + \langle \text{dist}(p, f) \rangle_j = \text{dist}(v, p) + \langle \text{dist}(p, S) \rangle_j = \text{dist}(v, p) + \beta_j^+(p). \]

However the last term is equal to \( \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \), which gives a contradiction to our premise \( \text{dist}'(v, S) > \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \).

So far we considered the case when the contribution of \( v \) to \( \tilde{C}_{t-1}(S_2) \) is given by a facility, or when the contribution of \( v \) to \( \tilde{C}_t(S) \) is given by an interface point. Thus the last case we consider is when the contribution of \( v \) to \( \tilde{C}_{t-1}(S_2) \) is given by an interface point, and the contribution of \( v \) to \( \tilde{C}_t(S) \) is given by a facility, i.e., \( \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} < \text{dist}'(v, S_2) \) and \( \text{dist}'(v, S) \leq \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \). We need to show that \( \text{dist}'(v, S) = \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \). First assume \( \text{dist}'(v, S) > \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \).

Due to Claim 23 this would imply \( \text{dist}'(v, S) > \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + d_j^+(p) \} \), which however contradicts our assumption to the contrary, i.e., that the contribution of \( v \) to \( \tilde{C}_t(S) \) is given by a facility. Hence we must instead have \( \text{dist}'(v, S) = \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \). According to Claim 22, our assumption that \( \text{dist}'(v, S) \leq \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + d_j^+(p) \} \) implies \( \text{dist}'(v, S_2) = \text{dist}'(v, S) \) or \( \text{dist}'(v, S) \geq \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \). In the former case, together with our assumption that the contribution of \( v \) to \( \tilde{C}_{t-1}(S_2) \) is given by an interface point, we would get \( \text{dist}'(v, S) > \min_{j \geq i, \ p \in I_j} \{ \text{dist}(v, p) + \beta_j^+(p) \} \), for which we saw above that this leads to a contradiction via Claim 23. Hence we are left with the other
implication of Claim 22, i.e., \( \text{dist}'(v, S) \geq \min_{j \geq i, p \in I_j} \{ \text{dist}(v, p) + \beta_j(p) \} \). This together with our conclusion from above, i.e., \( \text{dist}'(v, S) \leq \min_{j \geq i, p \in I_j} \{ \text{dist}(v, p) + \beta_j(p) \} \), means that the contribution of \( v \) to \( \hat{C}_\ell(S) \) and \( \hat{C}_{\ell-1}(S_2) \) is the same.

By analogous arguments, the contribution of any \( v \in B'_\ell \) to \( C_{B'_\ell}(S_1) \) is the same as its contribution to \( \hat{C}_{\ell}(S) \). Since \( B'_\ell \) and \( B'_{\ell-1} \) partition the set \( B'_{\ell-1} \), this means that \( \hat{C}_{\ell}(S) = C_{B'_\ell}(S_1) + \hat{C}_{\ell-1}(S_2) \), as required.

The next lemma implies that the compatible facility set minimizing \( \hat{C}_{\ell}(S) \) is considered as a solution when recursing over consistent distance functions.

\[ \textbf{Lemma 24} \quad \text{Let } S = B'_{\ell} \cap F \text{ be a facility set of } B'_{\ell} \text{ that is compatible with entry } \hat{T}[B'_{\ell}, |S|, (d_j^+, d_j^-)_{j=i}^\lambda(W)] \text{, and let } S_1 = S \cap B'_\ell \text{ and } S_2 = S \cap B'_{\ell-1}. \text{ Then there exist distance functions } (\delta^+_j, \delta^-_j)_{j=i}^\lambda(W) \text{ for } B'_\ell \text{, and } (\beta^+_j, \beta^-_j)_{j=i}^\lambda(W) \text{ for } B'_{\ell-1} \text{ such that }
\]

= \( (d_j^+, d_j^-)_{j=i}^\lambda(W) \), \( (\delta^+_j, \delta^-_j)_{j=i}^\lambda(W) \), and \( (\beta^+_j, \beta^-_j)_{j=i}^\lambda(W) \) are consistent, and

2. \( S_1 \) is compatible with entry \( T[B'_\ell, |S_1|, (\delta^+_j, \delta^-_j)_{j=i}^\lambda(W)] \) and \( S_2 \) is compatible with entry \( \hat{T}[B'_{\ell-1}, |S_2|, (\beta^+_j, \beta^-_j)_{j=i}^\lambda(W)] \).

To argue that the algorithm sets the value of \( \hat{T}[B'_{\ell}, S, (d_j^+, d_j^-)_{j=i}^\lambda(W)] \) correctly via (1), consider a set \( S \subseteq B'_{\ell} \) that is compatible with this entry and minimizes \( \hat{C}_{\ell}(S) \). By induction, Lemma 24 implies \( T[B'_\ell, |S_1|, (\delta^+_j, \delta^-_j)_{j=i}^\lambda(W)] \leq C_{B'_\ell}(S_1) \) and \( T[B'_{\ell-1}, |S_2|, (\beta^+_j, \beta^-_j)_{j=i}^\lambda(W)] \leq \hat{C}_{\ell-1}(S_2) \), where \( S_1 = S \cap B'_\ell \) and \( S_2 = S \cap B'_{\ell-1} \). From (1) we therefore obtain \( \hat{T}[B'_{\ell}, |S|, (d_j^+, d_j^-)_{j=i}^\lambda(W)] \leq C_{B'_\ell}(S_1) + \hat{C}_{\ell-1}(S_2) \). By Lemma 21 only compatible sets are stored in an entry by induction, and so the definition of \( S \) implies \( \hat{T}[B'_{\ell}, |S|, (d_j^+, d_j^-)_{j=i}^\lambda(W)] = \hat{C}_{\ell}(S) \), as required.

**Bounding the runtime.** To bound the size of the tables \( T \) and \( \hat{T} \), note that since there are \( \lambda(W) = \xi(W) + 1 \leq 2 \log_2(nX/\varepsilon) + 2 \) considered levels \( i \), and each level \( B_i \) of \( D \) is a partition of \( W \) where \( |W| \leq n \), there are at most \( O(n \log(nX/\varepsilon)) \) parts \( B \) considered by \( T \) in total. The other table \( \hat{T} \) considers the same number of parts, since a set \( B'_{\ell} \) can be uniquely mapped to the part \( B'_{i} \). The number of possible values for \( k' \) is \( k + 1 = O(n) \).

The domain \( \{ (x_j)_{0 \leq x < 2^{l+j}} \cup \{ \infty \} \} \) of a distance function for level \( j \) has at most \( 2^{l+j}/(\rho2^j) \) values, since \( (x_j) \) rounds a value to a multiple of \( \rho2^j \). The conciseness of the interface sets means that \( |H_B| \leq (h/\rho)^{O(1)} \) according to Lemma 5. Hence there are at most \( O(1/\rho)^{(h/\rho)^{O(1)}} = 2^{(h/\rho)^{O(1)}} \) possible distance functions. Since each entry of the table stores two distance functions for each of at most \( 2 \log_2(nX/\varepsilon) + 2 \) levels, the total number of entries of \( T \) and \( \hat{T} \) is at most

\[ O(n \log(nX/\varepsilon)) \cdot n \cdot (2^{(h/\rho)^{O(1)}})^{O(\log(nX/\varepsilon))} = (nX/\varepsilon)^{(h/\rho)^{O(1)}}. \]

Computing an entry of a table is dominated by (1). Going through all values \( k' \leq n \) and all possible consistent distance functions to compute (1), takes \( n \cdot 2^{(h/\rho)^{O(1)}} \) time, as there are \( 2^{(h/\rho)^{O(1)}} \) possible distance functions. Hence the total runtime is \( (nX/\varepsilon)^{(h/\rho)^{O(1)}} \), proving Lemma 20.

**The Facility Location\(^9\) problem.** To compute an optimum rounded interface-respecting solution to Facility Location\(^9\), the tables \( T \) and \( \hat{T} \) can ignore the number of open facilities \( k' \), i.e., they have respective entries \( T[B, (d_j^+, d_j^-)_{j=i+1}^\lambda(W)] \) and \( \hat{T}[B'_{\ell}, (d_j^+, d_j^-)_{j=i}^\lambda(W)] \).
Accordingly, compatibility of facility sets with entries is defined as before, but ignoring the sizes of the sets. The value stored in each entry now also takes the opening costs of facilities into account. That is, for any set of facilities $S \subseteq F \cap B$ in a part $B$ we define

$$C_B(S) = \sum_{v \in B} \chi_v(v) \cdot \min_{p \in I_p} \{ \text{dist}(v, S), \min_{j \geq i+1} \{ \text{dist}(v, p) + d_j^+(p) \} \} + \sum_{f \in S} w_f,$$

and an entry $T[B, (d_j^+, d_j^-)]_{i=1}^{\lambda(W)}$ stores the minimum value of $C_B(S)$ over all sets $S$ compatible with the entry, or $\infty$ if no such set exists. For $S \subseteq F \cap B_{\leq \ell}$ in a union of subparts $B_{\leq \ell}$ we define

$$\hat{C}_{\leq \ell}(S) = \sum_{v \in B_{\leq \ell}} \chi_v(v) \cdot \min_{p \in I_p} \{ \text{dist}(v, S), \min_{j \geq i+1} \{ \text{dist}(v, p) + d_j^+(p) \} \} + \sum_{f \in S} w_f,$$

and an entry $\hat{T}[B_{\leq \ell}, (d_j^+, d_j^-)]_{i=1}^{\lambda(W)}$ stores the minimum value of $\hat{C}_{\leq \ell}(S)$ over all sets $S$ compatible with the entry, or $\infty$ if no such set exists.

The entries of the tables can be computed in the same manner as before, but ignoring the sets sizes. In particular, the most involved recursion becomes

$$\hat{T}[B_{\leq \ell}, (d_j^+, d_j^-)]_{i=1}^{\lambda(W)} = \min \{ T[B_{\ell}, (\delta_j^+, \delta_j^-)]_{i=1}^{\lambda(W)} + \hat{T}[B'_{\leq \ell-1}, (\beta_j^+, \beta_j^-)]_{i=1}^{\lambda(W)} \mid (d_j^+, d_j^-)]_{i=1}^{\lambda(W)}, (\delta_j^+, \delta_j^-)]_{i=1}^{\lambda(W)}, (\beta_j^+, \beta_j^-)]_{i=1}^{\lambda(W)} \text{ are consistent} \},$$

Note that if $S_1 = B_{\ell} \cap F$ and $S_2 = B'_{\leq \ell-1} \cap F$ then these two sets are disjoint, and so $\sum_{f \in S} w_f = \sum_{f \in S_1} w_f + \sum_{f \in S_2} w_f$ for the union $S = S_1 \cup S_2$. Hence when proving $\hat{C}_{\leq \ell}(S) = C_B(S_1) + \hat{C}_{\leq \ell-1}(S_2)$ for Lemma 21, we can ignore the facility opening costs, and the proof remains the same as before. All other arguments carry over, and thus an optimum rounded interface-respecting solution for an instance of FACILITY LOCATION$^9$ can also be computed in $(nX/\varepsilon)^{(h/\rho)^{O(1)}}$ time.

## 5 Hardness for graphs of highway dimension 1

For both $k$-CLUSTERING$^9$ and FACILITY LOCATION$^9$ we present the same reduction from the NP-hard satisfiability problem (SAT), in which a boolean formula $\varphi$ in conjunctive normal form is given, and a satisfying assignment of its variables needs to be found.

For a given SAT formula $\varphi$ with $k$ variables and $\ell$ clauses we construct a graph $G_\varphi$ as follows. For each variable $x$ we introduce a path $P_x = (t_x, u_x, f_x)$ with two edges of length 1 each. The two endpoints $t_x$ and $f_x$ are facilities of $F$ and the additional vertex $u_x$ is a client, i.e., $\chi(u_x) = 1$. For each clause $C_i$, where $i \in [\ell]$, we introduce a vertex $v_i$ and add the edge $v_it_x$ for each variable $x$ such that $C_i$ contains $x$ as a positive literal, and we add the edge $v_if_x$ for each $x$ for which $C_i$ contains $x$ as a negative literal. Every edge incident to $v_i$ has length $(11c)^i$ for the constant $c > 4$ due to Definition 1, and $v_i$ is also a client, i.e., $\chi(v_i) = 1$. In case of FACILITY LOCATION$^9$, every facility $f \in F$ has cost $w_f = 1$, i.e., we construct an instance of the uniform version of the problem.

> **Lemma 25.** The constructed graph $G_\varphi$ has highway dimension 1.

**Proof.** Fix a scale $r > 0$ and let $i = \lfloor \log_{11c}(r/5) + 1 \rfloor$. Note that $\beta_w(cr)$ cannot contain any edge incident to a vertex $v_j$ for $j \geq i + 1$, since the length of every such edge is $(11c)^j \geq 11cr/5 > 2cr$ and the diameter of $\beta_w(cr)$ is at most $2cr$. Thus if $\beta_w(cr)$ contains a vertex $v_j$ for $j \geq i + 1$, then $\beta_w(cr)$ contains only $v_j$, and there is nothing to prove. Note
also that any path in $\beta_u(xr)$ that does not use $v_i$ has length at most $2 + \sum_{j=1}^{i-1} (2(11c)^j + 2)$, since any such path can contain at most two edges incident to a vertex $v_j$ and the paths $P_x$ of length 2 are connected only through edges incident to vertices $v_j$. The length of such a path is thus strictly shorter than

$$2 + 2 \left( \frac{(11c)^i}{11c - 1} - 1 \right) + 2i \leq 5(11c)^i - 1 \leq r,$$

where the first inequality holds since $i \geq 1$ and $c > 4$. Hence the only paths that need to be hit by hubs on scale $r$ are those passing through $v_i$, which can clearly be done using only one hub, namely $v_i$. ▶

To finish the reduction for $k$-CLUSTERING$^g$, we claim that there is a satisfying assignment for $\varphi$ if and only if there is a solution for $G_\varphi$ with cost at most $k + \sum_{i=1}^{\ell} (11c)^{iq}$. If there is a satisfying assignment for $\varphi$ we open each facility $t_x$ for variables $x$ that are set to true, and we open each facility $f_x$ for variables $x$ that are set to false. This opens exactly $k$ facilities and the cost of the solution is $k + \sum_{i=1}^{\ell} (11c)^{iq}$, since each of the $k$ vertices $u_k$ is assigned to either $t_x$ or $f_x$ at distance 1, and vertex $v_i$ is assigned to a vertex $t_x$ or $f_x$ at distance $(11c)^i$ that corresponds to a literal of $C_i$ that is true.

Conversely, assume there is a solution to $k$-CLUSTERING$^g$ of cost at most $k + \sum_{i=1}^{\ell} (11c)^{iq}$ in $G_\varphi$. Note that the minimum distance from any $u_k$ to a facility is 1, while the minimum distance from any $v_i$ to a facility is $(11c)^i$. Thus any solution must have cost at least $k + \sum_{i=1}^{\ell} (11c)^{iq}$, so that the assumed solution must open a facility at minimum distance for each client of $G_\varphi$. In particular, for each variable $x$, at least one of the facilities $t_x$ and $f_x$ is opened by the solution. Moreover, as only $k$ facilities can be opened and there are $k$ variables, exactly one of $t_x$ and $f_x$ is opened for each $x$. Thus the $k$-CLUSTERING$^g$ solution in $G_\varphi$ can be interpreted as an assignment for $\varphi$, where we set a variable $x$ to true if $t_x$ is opened, and we set it to false if $f_x$ is opened. Since also for each $v_i$ the solution opens a facility at minimum distance, there must be a variable in $C_i$ that is set so that its literal in $C_i$ is true, i.e., the assignment satisfies $\varphi$. Thus due to the above lemma bounding the highway dimension of $G_\varphi$, we obtain the Theorem 3 for $k$-CLUSTERING$^g$.

For FACILITY LOCATION$^g$ we claim that there is a satisfying assignment for $\varphi$ if and only if there is a solution for $G_\varphi$ of cost at most $2k + \sum_{i=1}^{\ell} (11c)^{iq}$. In fact the arguments are exactly the same as for $k$-CLUSTERING$^g$ above: if there is a satisfying assignment then a solution for FACILITY LOCATION$^g$ of cost $2k + \sum_{i=1}^{\ell} (11c)^{iq}$ exists, by opening the $k$ facilities corresponding to the assignment of cost 1 each. Conversely, any solution has cost at least $k + \sum_{i=1}^{\ell} (11c)^{iq}$ due to the edge lengths, and at least $k$ facilities need to be opened, one for each variable gadget. This gives a minimum cost of $2k + \sum_{i=1}^{\ell} (11c)^{iq}$, and any such solution corresponds to a satisfying assignment of $\varphi$. This proves Theorem 3 for uniform FACILITY LOCATION$^g$.

References

PTASs for Clustering in Low Highway Dimension Graphs


