Recent Development in Automatic Parameter Tuning for Metaheuristics

F. Dobslaw
Mid Sweden University, Information Technology and Media Department, Östersund, Sweden.

Abstract. Parameter tuning is an optimization problem with the objective of finding good static parameter settings before the execution of a metaheuristic on a problem at hand. The requirement of tuning multiple control parameters, combined with the stochastic nature of the algorithms, make parameter tuning a non-trivial problem. To make things worse, one parameter vector allowing the algorithm to solve all optimization problems to the best of its potential is verifiable non-existent, as can be inferred from the no free lunch theorem of optimization. Manual tuning can be conducted, with the drawback of being very time consuming and failure prone. Hence, means for automated parameter tuning are required. This paper serves as an overview about recent work within the field of automated parameter tuning.

Introduction

A common question when utilizing global optimizers on multidimensional and multimodal continuous or computationally hard (NP-) combinatorial optimization problems is, which initial parameter values to choose in order to yield (near-)optimal algorithm performance. This question is especially relevant within Constrained Programming, Artificial Intelligence, and Operations Research.

The majority of metaheuristics expose control to the user by at least a hand full of initial parameters. These parameters are either of numerical or categorical type. In order to yield good algorithmic performance do they have to be calibrated a-priori. The different types of parameters pose sundry difficulties in the analysis of their (combined) impact. CPLEX, for instance, a software for the solving of Mixed Integer Problems (MIP), allows for the setting of about 80 parameters, effecting the search and its result considerable.

Due to the large amount of studies utilizing metaheuristics on particular test problems, solid settings for those problems could be inferred. These parameter values are referred to as default or robust settings, because researchers found them to be suitable for the problem at hand. However, recent parameter tuning analyses reveal that these settings leave much space for improvement (see e.g., Hutter et al. [2010]), and that those parameter vectors are problem specific (Smit and Eiben [2010]), thus imply a high degree of probability for the algorithm to perform suboptimal on other problems. Still, many researchers do neglect those efforts, as a review on parameter usage for scientific contributions, utilizing Ant Colony Optimization (Wong and Komarudin [2008]), reveals.

Background

If there was an optimal setting for an optimization algorithm, converging towards the optimum on any optimization problem, the discussion about Parameter Tuning would be meaningless. Unfortunately, the no free lunch theorem of optimization reveals that there is no such thing as a globally best solver for all problems; at least in theory (Wolpert and Macready [1997]). One of the consequences of the no free lunch, having a huge impact on parameter tuning as a whole, is the fact that there is no best initial parameter setting for a metaheuristic for all optimization problems. Hence, optimal settings for initial parameters can vary significantly from problem to problem.
In the context of Evolutionary Algorithms (EA), Eiben et al. define parameter tuning as 'the commonly practiced approach that amounts to finding good values for the parameters before the run of the algorithm and then running the algorithm using these values, which remain fixed during the run.' (Eiben et al. [1999]). They further define parameter control as an alternative to parameter tuning, where initial parameter values are changing during the run of the algorithm. Parameter control is particularly of relevance when it comes to local search techniques, such as Tabu Search (TS), Self Organizing Maps (SOM), or Simulated Annealing (SA), where the dynamic change of the control parameters is build into the algorithmic design. Even though parameter control is a more complex problem, the border lines are somewhat fuzzy. Most of the dynamic behavior of a solver can be extracted as to extend the initial algorithm parameters. Examples are the lower and upper bounds for a parameter changing at runtime and the functional behavior by which it reaches from bound to bound (sigmoid, linear...). This is why separating the two attempts can be cumbersome. The focus of this paper, however, is on parameter tuning. The reason for this is that many interesting contributions have been published recently within the parameter tuning field only, describing methods that can be generally applied to any kind of solver.

Contributions presenting parameterless algorithms are not discussed here either. Usually, these parameter control methods are solver specific, and make assumptions about parameter values internally, without exposing them to the user.

**Automatic Parameter Tuning**

Because the task of parameter tuning can be very time consuming and tedious, an automated means to finding good initial parameter settings is desired. In Hutter et al. [2009b], the authors dub this unit, which itself is a (meta-)optimizer, the configurator of an algorithm. That is why they refer to parameter tuning as algorithm configuration. Figure 1 depicts the general idea. A target algorithm is becoming assessed on a variety of problem instances with different start conditions by a (semi-) automated configurator.

Currently, no standard approach supporting algorithm configuration exists, even though scientific contributions within the last years have led to a significant advancement of the field. Further, a standard data structure, as well as a common toolbox for parameter tuning do not exist either. For parameter tuning to become commonly utilized in industry and research, these are important requirements. The tools should be easy to apply, presenting statistically sound solutions without expecting the developer or user to possess an in depth knowledge about the configurator itself, parameter interactions, or the peculiarities of the problem (as mentioned in Ansótegui et al. [2009]). These are the reasons for why the objective of this paper is to give a general overview on the state of the art in parameter tuning.

In the remainder of the paper, parameter tuning and algorithm configuration are used syn-
DOBSLAW: RECENT DEVELOPMENT IN AUTOMATIC PARAMETER TUNING

onymously. After a short introduction into the different parameter types, does the following section present a variety of approaches to algorithm configuration. An overview of the experiments and achievements is given, and relevant results are reviewed. The final section concludes the paper, summarizing what has been done, and mentioning open research directions for future work.

**Parameter Types**

There are two types of target algorithm parameters to be optimized: numerical and categorical parameters.

**Numerical.** The numerical parameters can be further sub-categorized into integer and continuous parameters. Tuning numeric parameters is simpler in comparison to categorical parameters. That is the result of the natural distance metrics over the real and integer numbers, respectively. Here, heuristic search methods can be used to guide the search by the distance metric together with the assessed fitness from the target algorithm. Classic continuous optimizers, such as hill climbers, make use of gradient information for optimization. However, since we can not make any assumptions about the parameter space, are the continuous parameters not necessarily differentiable. Further, integer parameters are by no means differentiable. Gradient methods do not work for these kinds of MIPs (Mixed Integer Problems). That is why there is a need for non-gradient-based approaches.

**Categorical.** Categorical (or symbolic) parameters pose a higher level of complexity than numerical parameters do. They do not define any metric, thus cannot be ordered, and can therefore not be used as a means to guide a heuristic search from bad to improved regions of the parameter search space (interpolation is not possible). Each categorical parameter with \( n \) distinct values increases the size of the solution space for the parameter tuning problem by factor \( n \). Value elimination is more involved than for numerical values.

This is accompanied by the fact that categories can add so called sub-parameters to the initial parameter set, inducing that the comparison of categories requires many samples in order to spot parameter superiority; with a high likelihood there are settings that are better or worse for the distinct categories based on their sub-parameter choices.

To illustrate this, we consider the categorical parameter selection strategy for EAs. Utilizing tournament selection, two new sub-parameters appear; namely, the amount of competitors that are drawn randomly from the population and the amount of survivors of a competition (as mentioned in Smit and Eiben [2009]). Another selection method is the Boltzman-selector, introducing a sub-parameter temperature. Choosing Boltzman selection with a suitable starting temperature will beat tournament selection with an inappropriate amount of competitors, and vice versa. The question here is, which selection method is generally better in solving the problem, tournament or Boltzman?

**Approaches**

The objective of metaheuristics is to find good solutions in reasonable time. The involvement of randomness and the inspiration by biological or physical phenomena (e.g., flocking, annealing) lead to algorithmic behaviour which is difficult to trace and to analyze in order to draw sound, general conclusions about the algorithms properties; convergence for instance is very hard to prove. Because we are not able to foresee the interactions of parameters before the run of a metaheuristic on an optimization problem without trying it, an experimental approach to assessing the settings quality is inevitable (as already illustrated in Figure 1).

This paper distinguishes between two types of automated parameter tuning methods: model-free and model-based, a common distinction made in the literature (see for instance Hutter et al. [2010]). The main difference between the two is that model-based approaches build a model, interpreting the relation between the algorithm and its parameter values from the observations acquired by running the target algorithm with several settings; model-free approaches
don’t. Figure 2 gives an overview of the most influential methods in automated parameter tuning, to the best of the authors knowledge. It follows a brief historical walk through the approaches by category.

**Model-Free**

Model-free tuners draw implicit conclusions based on heuristic rules; choices for interesting parameter vectors to be investigate are often guided by randomness or a simple experimental design (e.g. Latin Hypercube Sampling, LHS). Usually they are more lightweight, and faster in execution, than model-based tuners. As a drawback, model-free tuners have very limited till extrapolation potential.

Finding good parameter values for an algorithm/problem combination is highly complex. This can be attributed to the fact that no assumptions about the problem landscape (dimensionality, multimodality, parameter interactions) can be made. For this type of problems a well-known population based method, Evolutionary Algorithm, is commonly applied. Since EAs are metaheuristics, their utilization can be seen as not solving the problem; they themselves have parameters to be tuned. However, they show effectiveness when improving target algorithm performance setup with default values.

Meta EAs, as first time introduced in Mercer and Sampson [1978], are EAs with the objective of optimizing metaheuristics. The EAs genome encodes the optimizers parameters. They are called meta EAs, since their activity can be considered meta optimization. The evolutionary approach intensifies good parameter regions, and supersedes bad ones. All EAs that can cope with real valued genome encoding can be used as meta EA for numerical parameters. A very common choice among EAs for this task is the CMA-ES (Covariance Matrix Adaptation - Evolutionary Strategies, see Hansen [2006]); ES has proven to be a good numerical optimizer. A relevant meta EA, the Meta-GA, was presented in Grefenstette [1986].

**Revac.** Relevance Estimation and Value Calibration of Parameters (REVAC) was presented in Mannen and Eiben [2006]. It estimates the distribution of promising parameter vectors by means of information theory. REVAC can be classified as an Estimation of Distribution Algorithm (EDA). It is limited to continuous parameter domains.

**ParamILS.** ParamILS is an abstract algorithmic description, presented in Hutter et al. [2009b]. The authors present two implementations: BasicILS and FocussedILS. Where BasicILS compares simple estimates for the cost statistics of subsequent runs, does FocussedILS attempt to overcome overconfidence based on the training instances by ‘adaptively choosing the number
of training instances to be used for the evaluation of each parameter configuration (Hutter et al. [2009b]). Both utilize iterated local search (ILS) techniques in order to guide the search towards promising areas in the search space. In short do both algorithms start out with one, user defined, initial parameter configuration. In succession sequentially combinations are assessed for performance, and the local search is conducted until a fitness threshold, or a time bound is reached. For each new experiment only single parameter changes are carried out. ParamILS requires the user to discretize all parameter ranges. An open source implementation of both algorithms is available\(^1\).

**Gender based Genetic Algorithm.** Gender based Genetic Algorithm (GGA, see Ansótegui et al. [2009]) is a GA utilizing gender differences. According to the authors, mate choice is more likely to have a high impact on result quality than, for instance, natural selection. Different kinds of selection pressure are utilized for the two genders. By only assessing one of the genders fitness, they report to safe half of the time, while producing an insignificant loss in performance. Due to the fact that only one gender is effected by selection pressure, does the other gender serve as a variety store.

The algorithm allows the user to enter variable or parameter trees to specify relatedness, if known in advance. These trees serve as the individuals genome structure. The method can cope with continuous and integer variables. The authors mention categorical variables, but the algorithm does, in its present version, not support them. A training set of problem instances can be specified for the algorithm; GGA randomly selects subsets of those, and races the individuals (parameter vectors) in each iteration against each other. The implementation makes use of parallel computations.

**Model-Based**

The essential objective for model-based tuners is to determine interesting sample points to be investigated, improving the model. A common tool here is the graphical interpretation of the parameter vectors with respect to the achieved target fitness. The related techniques are referred to as response surface methods (RSM); the umbrella term for this type of optimization is surrogate modeling. In other words, model-based approaches build models with the capability of 1) interpolating for the choice of new parameter settings to become investigated and even 2) extrapolating (recommending) good parameter vectors for new problems or problem instances.

Model-based parameter optimization methods have their roots in 1) traditional experimental design and 2) global black box optimization.

**DoE.** Design of Experiments (DoE) is a classic method for the conduct of representative experiments, firstly introduced in Fisher [1935]. It attempts to minimize the amount of experiments, while ensuring high quality results. Experiments have input variables (factors) and output variables (responds). DoE suggests extreme value, combined with center point analysis of the factor space. DoE can be utilized manually for parameter tuning, but was originally invented with the idea of physical experiments in mind.

**EGO.** In (Jones et al. [1998]), the authors propose an algorithm for the creation of a response surface model for black box functions. This algorithm makes use of Design and Analysis of Computer Experiments (DACE, Sacks et al. [1989]) for diversification and a branch and bound based phase for intensification of most promising areas. The algorithm is called Efficient Global Optimization (EGO). The authors demonstrate its quality in approximating the problem landscape of the Branin function by taking only very few sample points into consideration. The clue with this method is that it uses a so called *expected improvement* function with the objective to globally improve the model. The computation of this function is cheap and leads to proficient suggestions for further, promising, sample points to be assessed. EGO is restricted to deterministic algorithms or simulations. The authors thoroughly investigated surrogate model-

\(^{1}\)http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/
DOBSLAW: RECENT DEVELOPMENT IN AUTOMATIC PARAMETER TUNING

Figure 3. An EGO based parameter model, illustrated on a one dimensional function. The standard error of the model is drawn below the dotted line; high standard error implies high estimated improvement. The vertically drawn normal distribution at point $x$ represents the assumption on Gaussian distributed experimental noise. (from Jones et al. [1998])

ing and came up with a rule of thumb; in order to receive a good model should the amount of sample points be ten times the amount of problem space dimensions.

**F-Race.** F-Race is a, so called, *racing algorithm* (see Birattari et al. [2002]). It uses the Friedman test (two way analysis by ranks), a non-parametric statistical null hypothesis test. It consists of two phases: 1) the *aggregate* test, and 2) pair-wise comparison between individuals on a tournament basis. The population becomes reduced to a few best settings/individuals. F-Race assumes interactions of parameters to be linear. Due to scaling problems does the original version only work for a hand full of parameters. The reason for that is the expensive experimental design. Algorithm improvements, presented in Balaprakash et al. [2007], should allow for considerable more parameters to be optimized. F-Race can be downloaded: An extension of the mathematical programming language R called *race* is freely available\(^1\).

**Calibra.** The Calibra algorithm, proposed in Adenso-Díaz and Laguna [2006], combines experimental design (for exploration) and local search (for exploitation), in order to find good parameter settings. Because the underlying Taguchi approach is based on DoE, does Calibra inherit its scalability problems; a linear growing parameter space lets the amount of required experiments grow quadratic. The freely available Calibra tool \(^2\) can only be used for 5 parameters. It works for discrete and continuous parameters, requiring the continuous parameters to be discretized. Calibra can be applied to a whole set of training instances, in order to find robust parameter setting. A similar approach is presented in Coy et al. [2001].

**SKO.** Sequential Kriging Optimization (SKO, Huang et al. [2006]) is an algorithm that extends EGOs initial design for randomized algorithms or noisy functions. The algorithm bases the experimental design on simple Kriging, a collection of interpolation techniques for the creation of a response surface model. SKO restricts the user to optimizing continuous parameters. It is a sequential approach; the model is becoming updated each iteration until a certain quality or time-bound is reached.

**SPO.** Another sequential extension to EGO was presented in Bartz-Beielstein et al. [2005]. Gaussian distribution with mean zero variance was added to the model, representing experimental model error (in addition to experimental noise). Sequential Parameter Optimization (SPO) is not an algorithm but a methodology stressing the experimenter to ask the right questions in order to reduce the amount of experiments for the creation of a suitable model of the parameter space. The original version of SPO can handle continuous parameters only. Hutter et al. [2009a] extended the SPO framework by an initial LHS based experimental design, data

\(^1\)http://www.r-project.org/

\(^2\)http://coruxa.eupsig.uniovi.es/~adenso/file_d.html
log-transformation, and a new intensification mechanism as well as an expected improvement criterion. The extension resulted in a decrease in variation, and an improved model quality. The spot package for R, containing the initial version by Bartz-Beielstein, can be downloaded and used free of charge.

SMAC. In Hutter et al. [2010], the authors define a general template for parameter tuning which they call Sequential Model-Based Optimization (SMBO). It is composed of three repeated steps: 1) the model fitting, 2) the selection of new configurations to be investigated, and 3) a time-bounded intensification phase, in which the configurations are tested on a randomly selected subset from a user specified training set. Sequential Model-Based Algorithm Configuration (SMAC) is an implementation of SMBO. Training consists of fitness assessment based on 1) randomly distributed parameter settings, and 2) expected improvement. It is the first algorithm addressing multiple problem instance optimization, being able to cope with continuous, discrete, and categorical parameters likewise. Model interpolation utilizes a distance metric, based on the weighted Hamming distance and Random Forests, for categorical parameters. A Principal Component Analysis (PCA) is run in order to screen down the problem space. The authors consider it an aggressive racing algorithm, because it rejects inferior combinations online. A downloadable version was in working progress at the time this paper has been authored.

Techniques for Improvement

There are different techniques, improving target algorithm performance or tuning time, while optimizing parameters. Often, these techniques can be combined with existing approaches. The techniques presented here are called Racing, Sharpening, and Adaptive Capping.

Racing. Racing is used to aggressively reject parameter settings that are not competitive. For a randomized algorithm, sequences of experiments with equal settings are conducted, in order to prevent itself from the drawing of faulty conclusions on single lucky runs. Racing parameter vectors is the parallel running of two or more settings, comparing them online. Inferior ones are disregarded in future runs. Training time can be reduced significantly, since statistically inferior settings can be rejected before complete experimental series have finished. That is why racing is also referred to as an aggressive tuning method. Racing was introduced for the first time in Maron and Moore [1997].

Sharpening. SPO uses an intensification technique named sharpening (see Smit and Eiben [2009]). Sharpening shortens the traditional experimental each-vector-n-tests approach for randomized algorithms. It can be utilized for all kinds of tuning mechanisms. The main idea is that function evaluations are initially assessed only once for as long as the fitness is not considered especially promising (based on a threshold, e.g., local or global improvement). Once that threshold is reached, the amount of function evaluations for a vector is doubled, in order to avoid faulty conclusions about the goodness of parameter combinations.

Adaptive Capping. Adaptive Capping works similar to racing, cutting experimental series for inferior performing parameter settings on a problem (see Hutter et al. [2009b]). The authors present two implementations, namely trajectory preserving and aggressive capping. They demonstrate the improvements on three stochastic target algorithms, showing significant time improvements for series of experiments.

Experimental Review

In Grefenstette [1986], meta-GA was compared with default values, optimizing different algorithms on a set of numerical optimization problems. It yielded slightly better results than default settings. However, due to the lack of computational power at that time, experiments were too time consuming for serious consideration within the context of real world applications.

1http://www.cs.ubc.ca/labs/beta/Projects/AAC/index.html
In Bartz-Beielstein et al. [2005], the authors present a study on continuous benchmark functions (Sphere function, Rosenbrock, Rastrigin, Griewangk), optimizing a Particle Swarm Optimization (PSO) implementation. They report significant performance improvements in the order of multiple magnitudes for all except for the Griewangk function.

The Calibra algorithm had been shown in Adenso-Díaz and Laguna [2006] to improve the results for six scenarios, optimizing, for instance, Simulated Annealing and Tabu Search. It showed to improve the performance of the target algorithm, in some cases, significantly.

A study on tuning a basic GA, comparing meta-GA and REVAC, was presented in De Landgraaf et al. [2008]. Both optimizers improve performance quality, in some cases even considerable. REVAC showed in average to be slightly better than meta-GA on the testbed.

In Hutter et al. [2009b], the authors optimize parameters for three metaheuristics, including CPLEX, on different SAT encoded problems (e.g. graph coloring, and quasi group completion). The two ParamILS versions significantly improved the target algorithm performance compared to default parameter settings, in some cases with two orders of magnitude.

In Smit and Eiben [2009], the authors compare different meta-EA implementations with SPO for a standard EA on a 20 dimensional Rastrigin function. They utilize racing and sharpening for further improvements. The study reveals the superiority of CMA-ES and SPO to REVAC. A comparison with default initial parameter values was not undertaken.

ParamILS and GGA are compared in Ansótegui et al. [2009], optimizing three SAT solvers (SAPS, SAT4J and SAT4J*) on an instance of the SAT problem. The results reveal that GGA is competitive to ParamILS, with slight advantages for optimizing SAPS. Further, GGAs results are more robust, what the authors trace back to the fact that GAs are robust against noisy, multidimensional and multimodal functions.

In Hutter et al. [2009a], the authors investigated different versions of the SPO toolbox (SPO 0.3 and 0.4), comparing them to SKO and SPO+. The algorithm they optimized was CMA-ES on continuous test functions, such as Sphere, Achley, Griewank and Rastrin. The study reveals that SPO is more robust in its performance compared to SKO. Another comparison on SAPS with the same algorithms on a SAT problem instance shows that SPO+ outperforms the rest. It was concluded that the initial experimental design is not as relevant as the goodness of Gaussian distribution model. Lg-transformed data was recommended for analysis; it was found to give better model quality.

A more recent comparison of ParamILS, GGA and a new approach, called SMAC, depicts the superiority of the newcomer when applied to SAT problem solvers and CPLEX (see Hutter et al. [2010]). The work differentiates between training and test sets for the assessment of quality improvement for unseen instances. Another, model-free, approach was introduced as well (Random Online Aggressive Racing, ROAR), which showed its competitiveness in some, but not all of the scenarios.

In Smit and Eiben [2010], the authors optimize an ES via means of REVAC on several multidimensional functions (Ackley, Griewank, Sphere, Rastrigin, Rosenbrock). The results reveal that there are considerable differences between good parameter values for dissimilar functions. This result was especially valuable, since it supports the practical relevance of the no free lunch theorem of optimization.

The paper by Dobslaw [2010] discusses the usefulness of parameter tuning in general. Based on an experimental study does it show that the quality time trade-off can be very bad when considering default or robust parameter settings; in that case for discrete Particle Swarm Optimization (PSO) on the Traveling Salesman Problem (TSP). A simple DoE reveals that improved parameter settings can be found rather easily.

Discussion and Concluding Remarks

The article provided an overview of the development within the field of parameter tuning for metaheuristics in general. After a common introduction, a brief description about relevant
DOBSLAW: RECENT DEVELOPMENT IN AUTOMATIC PARAMETER TUNING

contributions was given, categorized into model-free and model-based. A review about conducted experimental studies, highlighting results in comparison and general observations about target algorithm improvement, was given. Completeness is not claimed, even though it was intended.

There are many open directions for future research. So far, there is only one very recently presented approach towards automated algorithm configuration of categorical parameter settings (Hutter et al. [2010]) to be found in the literature. First results look very promising, still other methods would be appreciated and are required for a comparison.

The extension of model-based approaches towards multiple instance learning should be further investigated. Correlations between initial parameter values and problem instance features could be analyzed in order to determine parameter settings for unseen problem instances. Single instance based training potentially leads to overfitting and at best to robust settings, yielding acceptable results on a large scale of problems, but good ones only for a few instances.

More studies on bio-inspired target algorithms should be conducted. Already published results could be reviewed and compared with the performance reached by the particular target algorithms under parameter tuning. Former studies were also very much directed towards the solving of SAT problems; a broader scope of optimization problems to be investigated would be of interest.

An open and well documented toolbox, allowing for simple usage and integration of new parameter tuning methods, is desirable. As mentioned above, there are some toolboxes out there, but their feature richness, documentation and usability could be improved. Additionally, a standard for defining the parameter tuning problem for applications should be specified. An XML (eXchange Modeling Language) based schema that can be read as input to the different toolboxes might be a good starting point. Such a toolbox could be used by practitioners as well as researchers in order for them to make use of the tuning methods, so to potentially increase solution quality and performance significantly while saving time; time to address the real research or industrial questions.

Another potential reason for why parameter tuning is not becoming applied on a big scale is the extra computational time for the initial experimental design and execution of the tuning process it implies. Field studies, showing the practical usefulness and reward in real applications, are required in order to convince others of its value; it’s return of investment.

The recent improvements indicate that automated parameter tuning is a relevant and prolific research area. It is desirable that the methods and algorithms frequently become applied in industry and research, so that the practitioners or researchers attention can be directed towards their real challenges.

References


