applied sciences

MDPI

Review

A Literature Survey on Offline Automatic Algorithm Configuration

Yasemin Eryoldaş * and Alptekin Durmuşoglu

Department of Industrial Engineering, Gaziantep University, Gaziantep 27010, Turkey; durmusoglu@gantep.edu.tr
* Correspondence: yasemineryoldas@gmail.com

Abstract: Metaheuristic and heuristic methods have many tunable parameters, and choosing their values can increase their ability to deal with hard optimization problems. Automated approaches for finding good parameter settings have attracted significant research and development efforts in the last few years. Because parameter tuning became commonly utilized in industry and research and there is a significant advancement in this area, a comprehensive review is an important requirement. Although there is very wide literature about algorithm configuration problems, a detailed survey analysis has not been conducted yet to the best of our knowledge. In this paper, we will briefly explain the automatic algorithm configuration problem and then survey the automated methods developed to handle this problem. After explaining the logic of these methods, we also argued about their main advantages and disadvantages to help researchers or practitioners select the best possible method for their specific problem. Moreover, some recommendations and possible future directions for this topic are provided as a conclusion.

Keywords: metaheuristics; offline algorithm configuration; parameter tuning

Citation: Eryoldaş, Y.; Durmuşoglu, A. A Literature Survey on Offline Automatic Algorithm Configuration. *Appl. Sci.* **2022**, *12*, 6316. https://doi.org/10.3390/app12136316

Academic Editor: Panagiotis G. Asteris

Received: 28 April 2022
Accepted: 16 June 2022
Published: 21 June 2022

1. Introduction

In recent years, due to the progress in modern sciences and technologies, there has been a fast increase in the size and difficulty of real-world continuous optimization problems (i.e., resource allocation, facility location, scheduling, vehicle routing, etc.). In many situations, these optimization problems must be considered a black-box problem, because there is not an obvious mathematical description of the problem, or in other words, because these problems are regarded as NP-hard; for obtaining optimal solutions using efficient algorithms, a lot of computing effort is required [1].

A metaheuristic can be defined as "a generic algorithmic template that can be used for finding high-quality solutions for a wide range of hard combinatorial optimization problems" by Birattari and Kacprzyk [2] and can be described as "higher level" heuristics. This topic is well studied and studies proved that metaheuristics (such as memetic algorithm, particle swarm optimization, or artificial bee colonies) offer near-optimal solutions to many types of optimization problems. The main differences between these algorithms are sourced from their searching patterns [3]. They can be used in a large number of research domains such as: telecommunications, machine learning, vehicle routing, etc.

Nevertheless, these metaheuristics have numerous specific parameters and design alternatives that considerably affect the efficiency and effectiveness of their performance. Additionally, these parameters need to be tuned to achieve the best performance of the algorithms. Although the selection of the best performing values for free algorithm parameters (we refer to this problem as "parameter tuning" or "algorithm configuration") is a challenging and tedious task, it can lead to an effective and good performing version of these algorithms. These algorithmic parameters are classified into two categories: categorical parameters (e.g., crossover operators in genetic algorithms) and numerical parameters (e.g., number of ants for ant colony optimization) [4].

Offline parameter tuning and online parameter tuning (or parameter control) approaches for choosing the best parameter values for optimization algorithms are suggested in the literature [5]. Parameter control approaches (which can be deterministic, adaptive, or self-adaptive [5]), first fix the initial parameter values and change them during the tuning process. However, in offline parameter tuning, parameter values are settled before the run of the algorithm and do not change during the process. In the scope of this study, we will use the terms "offline parameter tuning" and "algorithm configuration" with the same meaning. There is a considerable amount of research about offline parameter tuning methods that have been successfully applied to metaheuristics. In this study, we will focus on techniques for offline parameter tuning.

A parameter tuning problem can be defined in a simpler way, that is, given a parameterized target algorithm A, a space C of configurations, where each configuration c ϵ C specifies values for A's parameters, a distribution of problem instances I, and a performance measure m that measures the performance of A, the aim is to find a parameter configuration of A that provides optimal performance of A on I according to performance measure m [6]. Generally, the performance metric is based on the solution quality achieved within a given time budget or algorithm speed or variations and combinations of these two [6]. Eiben and Smit [4] stated that "algorithm robustness" can be used in conjunction with "performance measure" to assess algorithm quality. Algorithm robustness demonstrates the variance of algorithm performance across such factors: parameter values, problem instances, and random seeds [4]. For a formal and detailed definition of the algorithm configuration process, we refer to [2].

Until recently, the configuration of the parameters of these algorithms was carried out by the algorithm designer or end-user [7]. Because this ad-hoc, manual process is often boring and insufficient, automatic methods for finding appropriate parameter values have become available. These automatic methods lead to important improvements in the performance of target methods for solving different computationally hard optimization problems [8]. Because of its importance, a considerable number of automated parameter tuning techniques have been developed over the past few years, and parameter values found by these methods showed better results than default values identified based on human experts. Furthermore, studies have shown that using these algorithm configuration methods not only assists practitioners in identifying the best-performing parameter sets, but also in gaining a deeper understanding and analyzing the interaction between the target algorithm performance, problem instances, and selected parameter values [4]. Moreover, a successful tuning method will motivate algorithm designers to better parameterize their algorithms [9].

A comprehensive review of these studies is an important requirement because parameter tuning has become commonly utilized in the industry and academic community, and there has been a significant advancement in this area. Although there is a wide body of literature about the algorithm configuration problem, a detailed survey analysis has not been executed yet. This paper aims to give a general overview of the parameter tuning methods and shortly discuss applications of these offline algorithm configuration techniques. It also aims to discuss their main advantages and disadvantages to help researchers or practitioners select the best method for their specific problem. We also mentioned comparative studies of these methods, related research topics with the algorithm configuration problem, and other studies that consider different aspects of this problem. In the end, we provided recommendations and possible future directions for this topic.

This article is designed as follows; a brief introduction to the framework of the algorithm configuration problem and different parameter types is provided, and also a survey about offline algorithm configuration approaches and related works with these methods is provided. As a conclusion of the paper, we will summarize what has been conducted in this study, and we will mention open research directions for future work.

2. Literature Review

In recent years, parameter tuning has become commonly utilized in the industry and academic community, and there has been a significant advancement in this area. As a result of this research, there is a wide body of literature about the algorithm configuration problem, and a comprehensive review is an important requirement. In Bezerra et al. [10], authors reviewed two main topics, which are: the automatic configuration of multi-objective optimizers; and research about multi-objective configuration, which aims to optimize multiple performance metrics simultaneously to evaluate the algorithm performance, such as algorithm speed or quality of the solution. Hoos [11] provided a review of only three parameter tuning methods for metaheuristics and gave information about the applications of these methods.

Algorithm configuration methods have been classified in many different ways in the literature. Eiben and Smit [4] is the most detailed survey, in which they constructed three different taxonomies to distinguish and categorize algorithm configuration techniques. In their study, the parameter tuning methods are divided into four categories, such as: meta-EAs, sampling methods, screening, and model-based methods. In [12], they classified tuning methods as tuning by analogy, hand-made tuning, hybrid tuning, experimental design-based tuning, and search-based tuning. Huang et al. [13] conducted a review of automatic algorithm configuration techniques and classified these methods as simple generate-evaluate techniques, iterative generate-evaluate techniques, and high-level generate-evaluate techniques based on the structure of these configuration methods. Schede et al. [14] provided classification schemes for algorithm configuration problems and for algorithm configurators. Additionally, they summarized the related literature on these topics.

In this paper, we will use a more general distinction of automated parameter tuning methods made in the literature: model-free and model-based tuning approaches (see [6]) with some modifications. We will describe racing methods and traditional experimental design-based methods as separate categories. In this paper, we examined the developed automatic algorithm configuration methods concerning this categorization. Additionally, for explaining each method (especially for widely used methods, due to the space limitation), we used some criteria such as search strategy, parameter type, expected output, stop criterion, and configuration objective (multi or single). Additionally, we also mentioned whether these methods use surrogate models or not.

Figure 1 demonstrates the classification of algorithm configuration methods and an overview of the most widely used automated algorithm configuration methods in each class.

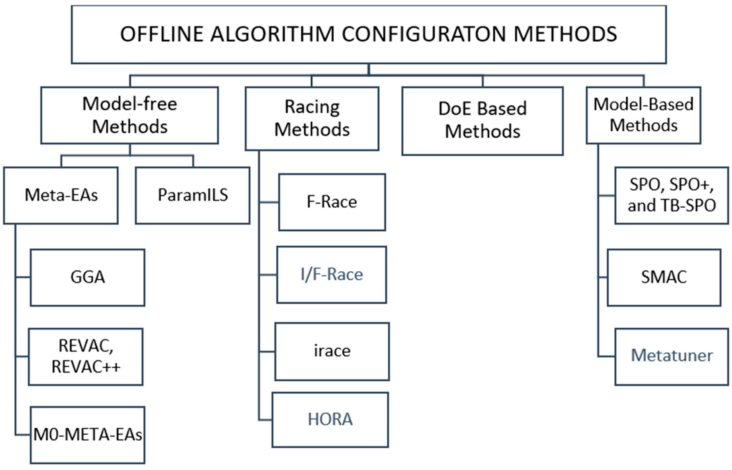

Figure 1. An illustration of the classification of automated algorithm configuration methods.

2.1. Model-Free Algorithm Configuration Methods

Model-free methods are generally based on heuristic rules, and choices for parameter vectors to be investigated are often selected by randomness or experimental design techniques. Although they are faster in execution than model-based tuners, they have very limited extrapolation potential. Research on this topic started in the early 1990s [15,16] and has lately been gaining momentum. Meta-EAs and ParamILS are in the scope of model-free algorithm configuration methods.

2.1.1. Meta-EAs

We previously mentioned the difficulties of finding good parameter values (including conditional and categorical parameters) for heuristics and metaheuristics. This is a complex stochastic black-box mixed-variables optimization problem due to interacting variables, nonlinear objective function, multiple local optima, lack of analytic solvers, and noise [4]. Ironically, in this type of problem, heuristic optimization algorithms are effective solvers and can find good parameter settings [13].

Because of this reason, using any type of evolutionary method to configure the parameters of another evolutionary algorithm is a natural opinion. They are called Meta EAs because their activity can be considered meta optimization. Mercer and Sampson [17], and Grefenstette [18] are the first study that aimed to use an Evolutionary Algorithm for optimizing metaheuristics. After these studies, research continued and several studies were conducted about meta-GAs. For details of studies on Meta-GA and critiques about the use of meta algorithms for investigating the best parameter vectors, readers are suggested to see [19].

One advantage of meta-EAs is that they can be terminated at any point of the tuning process and give a solution at that point. However, they are not suitable for large parameter spaces, because they can provide sustainable solutions only after a large number of evaluations of candidate configurations [13]. Additionally, most of the meta-EAs (i.e., CMA-ES and REVAC) cannot deal with the categorical parameters [13].

GGA

Ansótegui et al. [9] proposed a naturally parallel gender-based genetic algorithm (GGA). Because genetic algorithms are inherently parallel, racing them against each other makes evaluating competitive individuals (the most time-consuming part of algorithm configuration) easier. In other terms, a tournament-based selection mechanism is used for testing the competitive population. They introduced the gender separation method to reduce the high evaluation cost for the genetic algorithm and apply different selection pressure to both genders. Additionally, new candidate configurations are generated by using different sub-populations and a specialized cross-over operator. GGA terminates when the runtime limit is achieved or when the average performance of the population is not further improved [14]. According to their report, parallel evaluation of configurations supports the configuration goal of minimizing algorithm run-time while producing an insignificant loss in performance. For the improvement of this study, they recommend selecting the most important component(s) of GGA for its performance. It can deal with both discrete and continuous parameters and give one best parameter configuration as an output. The GGA method is widely used in tuning literature, for example, in tuning solvers in SAT and mixed-integer programming [20], Max-SAT [21], and machine reassignment [22].

GGA is extended to GGA++ by using surrogate models with population-based approaches in the study [23]. In GGA++, instead of using random recombination, they developed specialized surrogate models based on random forests and combined them with a genetic algorithm for finding the most promising offspring. After the experiments, model-based GGA outperformed other configurators such as GGA and SMAC for tuning two SAT solvers. Ansótegui et al. [24] suggested a Python tool that implements a distributed version of the GGA, PyDGGA, which maximizes the usage of parallel resources by simulating

future generations of the genetic algorithm even before the current generation is completely finished [24].

REVAC

Nannen and Eiben [25,26], suggested Relevance Estimation and Value Calibration of parameters (REVAC), and enhanced meta-EAs, to tune numerical parameters of optimization algorithms (continuous and categorical parameters needs to be discretized). The REVAC approach is based on EDA (Estimation of Distribution Algorithm [27]). Revac starts with the "M" configurations and at each iteration, the oldest configuration is changed with a newly created child parameter vector with the aid of multi-parent crossover and a mutation transformation. Information theory is used to estimate the expected performance when parameter values are chosen from a probability distribution (parameter relevance) instead of predicting the algorithm performance for different parameter configurations. For each tunable parameter, it gives one best value and one interval of values. It stops when the assessed number of execution is achieved. According to the Shannon entropy measurement of these distributions, a parameter that shows low entropy (which means the parameter value range is narrow) is considered more relevant to the algorithm performance. For a detailed discussion about REVAC's procedures details such as probability distributions and entropy, [28] is a good guide. In [19], a comparison of REVAC and Meta-GA was executed, and REVAC found better performing and robust configurations than those configured manually and by the meta-GA.

Nannen et al. [29] executed a study about the effects of different design choices that underline the cost of tuning. The REVAC method was further enhanced in the study of Smit and Eiben [30] by using an advanced technique of aggregating the performance of multiple runs. Another extension of REVAC was introduced by Smit and Eiben [31] to find widely applicable parameter values. According to the no-free-lunch theorem [32], a parameter configuration that performs well on one kind of problem can perform worse on the other kind. Smit and Eiben [31] executed the study in order to find good parameter values of a simple genetic algorithm for more than one widely used test function with REVAC tuning. So, they compared their generalist EA (whose parameters are tuned for a set of test functions) with a benchmark EA (whose parameter values are chosen by 'common wisdom') and with a specialist EA (which is a parameter set that shows good performance on one specific type of problem). The term "generalist" is used to demonstrate parameter values that perform well on a whole possible set of test problems, and this is infeasible according to the no-free-lunch theorem. Therefore, their claims turned out to be that generalist parameter values perform well on the previously defined set of test functions. They described this complete system in MOBAT [33] (available at http://mobat.sourceforge.net, (accessed on 10 June 2022)). Results of the experiment demonstrated that REVAC is also able to find good parameter vectors for a set of test problems (not only for a single problem), which are different from the suggested values by common wisdom. They underline that the best generalist will always be linked with the definition of the generalist. Furthermore, this experiment demonstrated that several runs per parameter vector are critical, where excessive runs have high computational costs and too few runs lead to inaccurately estimated utility and inaccurate results. Additionally, they mentioned specific problems raised in a generalist configuration process based on the difference in difficulty levels of problems in the test suites, which cause a hidden bias. They suggested using a method from multi-objective optimization, aiming to establish the Pareto front to overcome these problems.

Multi-Objective META-EAs

As algorithm configuration procedures improved, researchers focused on the multi-objective algorithm configuration problem instead of optimizing a single performance objective, because in many cases, multiple competing performance objectives have importance for algorithm configuration problem, such as running time, memory consumption,

solution quality, etc. A multi-objective automatic algorithm configuration problem also aims to explore the trade-offs between multiple performance objectives (the initial studies on this topic are [34–37]). We will summarize the multi-objective algorithm configuration methods that take inspiration from multi-objective evolutionary algorithms (MOEAs) in this section.

As mentioned in the study [31], when more than one problem or performance metric is used, an easier method is to aggregate them into one measure for optimizing all of them. However, according to the results of this study, such an approach leads to several problems. An alternative method to aggregation is regarding each performance measure and test function in the test set as one objective [38], which means creating a parameter Pareto front (non-dominated parameter sets, satisfying more than one objective).

In the method of Dréo [39], they used multiple performance measures (algorithm speed and algorithm accuracy) of four different algorithms at the same time without specifying weights for those two objectives beforehand. However, they handle only one parameter at a time, and the test suite is a unique problem. They used the multiple runs (fixed number) and averaged them to estimate the utility. A multi-objective optimization algorithm, NSGA-II, has been used as a tuner. Although using performance fronts is costly, their results prove that it is very suitable for the parameter configuration problem of metaheuristics.

Additionally, Smit et al. [38] extended the study of [39] to a multi-function optimization problem, where each fitness function in the problem set represents one objective (so the number of objectives is the same as the number of fitness functions). They introduced the Multi-Function Evolutionary Tuning Algorithm (M-FETA), which is based on a Multi-Objective Evolutionary Algorithm (MOEA) method and can approximate the parameter Pareto front. The parameter Pareto set includes non-dominated parameter vectors, which means that all other parameter vectors perform significantly worse. In such configuration problems, the utility of parameter vectors is defined by the performance of the algorithm (with these parameter values) on a collection of functions. Because heuristic optimization algorithms are stochastic, the performance of a parameter configuration can only be estimated due to the noise in the results. A common way of improving these estimates is by repeating the measurement. However, it means doing more executions, and so it is a costly way of gaining more confidence. M-FETA has a special technique for assessing the quality of candidate parameter vectors. In their technique, the number of expensive tests is reduced (or confidence improved) by judging the utilities of neighboring parameter vectors evaluated before. Their results showed that Pareto fronts allow for the investigation of interactions between parameter stings, fitness functions, and the optimization algorithm [38]. When compared to the single objective meta-EA, these approaches provide an important added value.

Following the same idea, BONESA [40] is suggested as an iterative model-based approach to find a robust parameter set for multiple test instances. According to their experimental results, their method can provide a good estimation of the utility values of parameters and collects a lot of useful information on EA robustness. Moreover, they developed a tool for analyzing and giving insight into EAs (available at https://sourceforge.net/projects/tuning/ (accessed on 10 June 2022)).

As a recent example of meta-EA, Ugolotti and Cagnoni [41], proposed EMOPaT (Evolutionary Multi-Objective Parameter Tuning), which uses a multi-objective optimization algorithm (NSGA-II [42]) for parameter configuration of differential evolution (DE) and particle swarm optimization (PSO) algorithms. The best parameter configuration for a given problem is selected based on a multi-objective approach, which aims to maximize the algorithm performance for a given problem set. They also summarized algorithm configuration studies that used meta-EAs. Ugolotti et al. [43] evaluated EMOPaT on different functions and showed that it cannot only find good parameter vectors for the training problems but also extracts new parameter vectors that demonstrate good performance on unseen test problems from those results.

Dymond et al. [44] suggested the tMOPSO (tuning multi-objective particle swarm optimization) method for tuning an algorithm based on a bi-objective performance measure, which is the best objective function value found and the number of objective function evaluations (OFEs). Dymond et al. [45] suggested MOTA, a many-objective tuning algorithm, particularly for configuring a stochastic optimization algorithm based on multiple performance measures. Experimental results of the configuration process showed the effectiveness of MOTA for many-objective tuning.

2.1.2. ParamILS

ParamILS, another model-free method, suggested by [46] and improved by [47], is an iterated local search algorithm, which is influenced by the hand-made algorithm configuration methods. The ParamILS process begins with a default parameter value generally based on user experience with randomly chosen r configurations from the dedicated configuration space, and the performance of the algorithm is improved iteratively by searching in its neighborhood, which can be described as changing only one parameter. In other words, an iterative first-improvement algorithm is a local search procedure of ParamILS. These selected (r + 1) configurations are evaluated and the local search process is initialized with the best performing configuration. The ParamILS builds a chain of local optima by iterating the following steps. Starting from the initial solution, it performs s random perturbation to avoid local optima, which are implemented via changing the values of a few parameters to randomly chosen values. Then a local search process is performed and a comparison is executed between the newly obtained candidate configuration and the previous best one to decide whether to keep or reject the new one. According to this comparison, if an improvement on the best solution is observed, the new one is selected as the best configuration. Additionally, ParamILS also includes a diversification mechanism with a re-start probability (p_r). After each local search, ParamILS is restarted using a random initial configuration. It stops when the allowed number of executions or time limit is achieved and, it gives one best performing configuration as an output. In their empirical study, for all tuning scenarios, parameter configurations found by ParamILS always demonstrated better performance than the default parameter values and the CALIBRA system [48].

Two different versions of ParamILS (available at http://www.cs.ubc.ca/labs/beta/Projects/ParamILS/ (accessed on 10 June 2022)) are suggested, such as BasicILS and FocusedILS. Their differences are in the way of defining one configuration as better than another one. BasicILS uses a fixed number of random seeds or (problem instance) for performance comparison among the configurations, while FocusedILS uses the dominance concept. Essentially, one configuration dominates the other (i.e., c_1 dominates c_2 configuration) if and only if the mean performance of the algorithm with c_1 on m_1 seed is better than c_2 on m_2 seeds when $m_1 > m_2$. In other terms, when FocusedILS finds that configuration c_1 performs better than c_2, it executes additional runs (by increasing the number of seeds or instances to be solved) to prove that 'c_1 dominates c_2'. Moreover, although over-tuning and over-confidence problems were already argued by [2], the first statistical arguments and experimental results are given in this study. For details of these problems and a detailed definition of a better mechanism, we refer to the original paper [46].

ParamILS method is improved by introducing a new "*adaptive capping*" mechanism [47]. The results of their experiments showed that adaptive capping accelerates both BasicILS and FocusedILS and can find well-performing parameter settings of complex and highly parameterized algorithms such as the commercial optimization tool CPLEX (available at http://www.ibm.com/software/integration/optimization/cplex-optimizer/ (accessed on 10 June 2022)). The idea behind adaptive capping is to avoid redundant runs of the target algorithm by developing bounds on the performance measure to be optimized. Cáceres and Stützle [49] investigated variable neighborhood search mechanisms instead of the local search phase of ParamILS (one-exchange neighborhood), and they obtained promising results.

Further to this, multi-objective extensions of BasicILS and FocusedILS are introduced in [37]. MO-ParamILS uses a multi-objective iterated local search procedure. In this extended method, a non-dominated configuration set is modified iteratively. Additionally, this property differs from the single objective ParamILS. They used MO-ParamILS to demonstrate trade-offs between solution quality and running time performance measures, and secondly between memory usage and running time. Experiments demonstrated that both are effective tools for bi-objective configuration cases, but MO-FocusedILS outperformed MO-BasicILS. For future research, they offer to apply the multi-objective configuration to multi-objective algorithms, which are difficult to configure and to develop multi-objective extensions of other automatic configurators. In the next study, Blot et al. In [50], they applied the MO-ParamILS to configure a multi-objective local search algorithm applied to bi-objective permutation flow shop problems. They extended their experimental study in [51] by considering the bi-objective TSP problem and also considering large instance sets. In [52], the authors discussed the effect of correlation between optimization objectives on algorithm configuration for multi-objective algorithms, and their results proved that the performance of these algorithm configuration methods is not affected by the degree of correlation between objectives. However, the correlation between objectives and the problem set size affects the best performing configuration.

One aspect of ParamILS is that it can configure only categorical parameters; otherwise, it requires discretizing numerical parameters. This can be considered a disadvantage of ParamILS, but different configuration problems, such as [53,54], demonstrate that this is not a major obstacle to the performance of ParamILS. In the above, we provided a basic overview of applications of ParamILS, but for details of other ParamILS applications, we refer to [11,47].

We summarized the main advantages and disadvantages of commonly used model-free algorithm configuration methods in Table 1. It must be stated that the advantages and disadvantages of the offline tuning methods that we summarized in Tables 1–3 are related to their specific underlying principles (such as the elimination method or search space) and their specific assumptions. While these advantages and disadvantages are method-specific and in different domains, we cannot present them in a correlation between the different methods. This could only be performed under very restricted pre-defined criteria, but in this case, it does not reflect the total contribution or limitations of the mentioned methods.

Table 1. A summary of the advantages and disadvantages of widely-used model-free algorithm configuration methods.

Methods	Advantages	Disadvantages
RAVAC++	• Using the entropy value of a parameter is very precious because it demonstrates how much tuning effort each parameter requires [28]. • REVAC gives an interval for parameter values as a final report, and it can be an advantage only when this interval is narrow [55]. • REVAC is enforced with racing and sharpening methods, and hence it can efficiently handle the stochasticity of the utility values and can be used for examining the cost of tuning and also examining the relevance and sensitivity of the parameters [29].	• One of the weaknesses of REVAC, it is required to choose a random value from the returned interval for the parameter and it may be an inappropriate value. Additionally, so, performance differences between individual configurations cannot be demonstrated exactly. • Another drawback of the Revac is that it cannot tune categorical parameters [13,28].

Table 1. *Cont.*

Methods	Advantages	Disadvantages
GGA++	• GGA++ is among the most competitive and robust tuners suggested. • The main advances of GGA++ are its tree-based representation and used operators that can handle any type of parameters (both numerical and categorical parameters without discretization) simply [56]. • Racing, sharpening, and capping methods are provided with it [56]. • GGA++ uses an adaptive capping mechanism like ParamILS, and it combines it with a parallelization mechanism to use multiple processing units efficiently [56].	• GGA++ could not deal with the complex parameter types (such as conditional parameters) that can occur in some solvers [56].
ParamILS	• It is a sophisticated method and can be used for tuning various algorithms, even if they have many parameters (categorical and numerical), and always demonstrated a very good performance in the vast majority of cases studied [47]. • FocusedILS can effectively deal with the over-turning and over-confidence problem [46]. • The main advances of this method are using racing, sharpening, and capping. Adaptive capping can seriously accelerate the elimination process of poor performing configurations, so increase its performance [47].	• Because ParamILS is based on stochastic local search, the selected random seed can affect its performance. • Major drawback of ParamILS is that it can only deal with discrete parameters and so it can be uninformative in terms of alternative parameter vectors and the parameter's space structure. Furthermore, it is difficult to analyze interactions between parameters, specific instances, and parameter configuration correlations [14,47]. • Capping mechanism cannot be efficient when the performance criterion is solution quality [13].

2.2. Racing Methods

2.2.1. F-Race

Birittari et al. [57] introduced an automatic racing procedure for automatic algorithm configuration problem based on the methods from the machine learning literature for model selection [58,59] and Friedman's nonparametric two-way analysis of variance by ranks [60]. The main difference between the racing methods and the system selection methods is that the racing methods deal with a larger set of competing systems when compared with the system selection methods.

This procedure aims to find the best-performing parameter vector of a target algorithm from a finite configuration set by eliminating bad-performing ones whenever they are proven to be poor statistically, and the procedure is iterated over the remaining ones. The F-Race process starts with performing n runs for each configuration to acquire sufficient data before elimination. Obtained data are recorded for each parameter configuration. At each iteration, a problem instance is randomly selected for the performance comparison of candidate parameter configurations. After evaluating configurations on every instance, costs (or utilities) of each are added to the stored values. After each iteration, Friedman's two-way analysis of variance by ranks, a nonparametric statistical test, is used to control if there are significant differences among the configurations. If the null hypothesis is rejected with a significance level, it can be inferred that at least one parameter configuration is better than at least one other parameter configuration. After that situation, a pairwise comparison between parameter configurations is performed. Statistically poor configurations are discarded from the candidate's configuration set. The termination condition is either a remaining one parameter vector or reaching a previously defined time budget, and it gives a best performing configuration set as an output. A block design based on different problem

instances (one instance considered as a block) is applied by F-Race. The null hypothesis of the Friedman test claims that all possible rankings of the parameter vectors within each block are equal.

Some limitations of this method are that discretization is needed when tuning continuous parameters, and another important limitation of F-Race (available at https://cran.r-project.org/src/contrib/Archive/race/ (accessed on 10 June 2022)) is that at the beginning of the process, all configurations have to be evaluated. Because of this limitation, published experiments with F-Race have evaluated a maximum of 1200 configurations. The advantages of this method are that the elimination of worse configurations accelerates the procedure and leads to a more reliable evaluation of the promising ones. Additionally, an adaptation of blocking design to general racing methods is the first in the nonparametric test. This method fills the gap between the Hoeffding race [59], which is a nonparametric test, and BRACE [59], which considers a blocking design.

2.2.2. I/F-Race

To overcome the stated limitation of F-Race method, Balaprakash et al. [61] have developed Sampling and Iterative F-Race, which are based on the previously developed F-Race method. Although this F-race can deal with large configuration spaces, it handles only numerical parameters. Birattari et al. [62] extended this work to handle categorical parameters. For the extension of I/F-Race, they discuss design issues such as the number of iterations, the number of candidate configurations, the computational budget for each iteration, the termination procedure, and how to generate candidate configurations. The computational budget is distributed equally over these iterations and a different sampling strategy is suggested. As a conclusion of these discussions, they proposed a different version of the iterated F-Race algorithm and defined an extended iterated F-Race based on the procedure suggested by [61]. According to their experimental results, I/F-Race demonstrated better results in each experiment than F-Race, which uses a full factorial design, F-Race (FFD), and F-Race (RSD). F-Race is one of the most widely used algorithm configuration methods in many different application areas. For a survey of studies that used F-Race in different areas, we refer to the paper [62].

2.2.3. Irace

López-Ibáñez et al. [7] described the irace package, which implements the improved and extended version of the iterated F-race. They introduced recent extensions such as a restart mechanism, the use of truncated sampling distributions, and an elitist racing that tests parameter vectors on a problem set whose size is increased in every iteration of irace. Finally, they experimentally evaluated this recent version of irace on two configuration cases. It has been underlined that the parameter sampling method and the statistical assessment of the performance of different configurations have vital importance for guiding the iterated racing procedures to find the better configurations. Zhang et al. [63] is one of the recent studies that use irace to configure the newly designed multi-objective evolutionary algorithm. For an overview of the various and wide range of applications of the irace package (available at https://iridia.ulb.ac.be/irace/ (accessed on 10 June 2022)) in different studies, we refer to [7].

The irace was developed for optimizing the obtained solution quality for a defined running time. However, when used to optimize the running time, irace cannot perform as well as other configuration approaches. The reason is that it spends too much time evaluating poor configurations. Cáceres et al. [64] aimed to improve the performance of the irace method for running time minimization objective, and they added an adaptive capping method to the irace as ParamILS uses. Additionally, the suggested new irace uses a new technique, dominance elimination, which rejects poorly performing configurations. They showed that these improvements over irace lead to a high-performing irace that is competitive with state-of-the-art algorithm configuration methods (ParamILS and SMAC)

whose objective is minimizing the running time. They also deeply analyzed the behavior of irace and compared different methods of integrating adaptive capping.

Cáceres, Bischl, and Stützle [65] investigated the use of random forests models, a common improvement technique for search methods when evaluations are computationally hard, as surrogates in irace. The usage of the random forest model is different from that used in SMAC and more similar to the use in GGA++.

2.2.4. HORA

Barbosa et al. [66] proposed an algorithm configuration method (HORA) based on combining the racing procedure with the design of experiments (DOE) techniques. Although this method is similar to F-Race, new candidate configurations are created based on the neighborhoods of good parameter vectors, and DOE is used to create the initial configuration set. It can deal with small search spaces and adaptive capping is not performed during races [14]. Four parameters of the genetic algorithm and simulated annealing algorithms are tuned with this method. For the problem layer, the traveling salesman problem (TSP) and the scheduling problem to minimize the total weighted tardiness in a single machine (TWTP) problem are used. According to results obtained from experiments, the target algorithms' performance is improved with tuning HORA, and although their method is promising and fast, more studies must be executed to confirm its effectiveness.

2.3. Design of Experiment Based Methods

Many optimization algorithms, such as metaheuristics or commercial solvers, like the CPLEX solver with 76 parameters [54], contain a large set of parameters. In these situations, decreasing parameter search space is needed to efficiently apply automated tuning procedures. One way of achieving this reduction is decomposing the parameters into disjoint partitions and configuring these parameters one by one.

Design of experiments (DOE) [67] is another method for decomposing parameter space. Design of experiments techniques have been widely used to manually determine the best parameter setting of metaheuristics [68–74]. We will give a brief overview of the main studies that use the design of experiment-based techniques for the selection of parameter values of optimization algorithms.

Coy et al. [75] suggested a widely applicable and effective method for optimizing continuous parameters. Their approach consists of a full factorial design and a gradient descent method. They evaluated their approach to two new vehicle routing heuristics for solving different capacity and route-length constrained vehicle routing problems. The inconvenience of their methods is that it approximates the response surface linearly and does not investigate the relationship between instances, parameter vector, and algorithm performance.

Adenso-Diaz and Laguna [48] proposed the CALIBRA system that starts with Taguchi's fractional factorial design and then applies a local search procedure. In the beginning, CALIBRA tests each parameter vector with two levels of the full factorial design per parameter. Then, at each iteration of the applied local search procedure, it iteratively approximates regions of promising parameter configuration in the configuration space. The local search procedure evaluates nine configurations around the existing best-performing configuration. This process is terminated when local optimum criteria have been reached. A local search procedure can be executed more than one time if the computational budget remains to obtain a more local optimal parameter configuration. The experiment is executed on six existing heuristic-based methods. Although the results of the experiment demonstrated that CALIBRA is a useful tool for the algorithm configuration problem, there are some limitations, such as it can only handle five numeric and ordinal parameters, it does not guarantee optimality, and the local search procedure cannot operate the effects of interactions between parameters.

Akbaripour and Masehian [3] proposed an approach for algorithm configuration based on a combination of design of experiments (DOE), signal to noise (S/N) ratio, Shannon entropy, and VIKOR methods. Their approach aims to optimize four goals: solution

quality, several fitness function evaluations, minimizing the algorithm's runtime, and variance of these objectives. According to the results of their experiment, the suggested approach improved the average number of iterations, average running time, and algorithm's solution quality.

Dobslaw [76] suggests a parameter tuning technique, which is a combination of design of experiments and artificial neural networks (ANN). Their methodology consists of four phases: problem description, training, parameter retrieval, and execution. The first phase includes a computer-readable representation of the process (problem, quality measure, and target algorithm). In the first part of the training phase, an automated design of experimental procedure is applied to a finite training set of problem instances for improving the initial values of the parameter. Later, the outcomes of the first part are used for training an artificial neural network in order to suggest near-optimal initial parameters for any given problem instance. In the third phase, ANN is used as a predictor to receive promising initial parameter sets for any given problem. As a final stage, the algorithm is executed with the suggested values from phase three. They tuned the standard PSO 2007 metaheuristic for solving the traveling salesman problem (TSP). The proposed parameter tuning methodology can be used to solve any configuration problem.

Pham [77] proposed an algorithm configuration method that combines factorial design and fuzzy logic, aiming to compromise conflicting demands such as convergence speed, robustness, and versatility (applicability to many different problems) for the dynamic optimization of chemical processes by a highly parameterized evolutionary algorithm. Their approach starts with a factorial experiment, and then average parameter levels for the best and worst runs in each problem are determined. Then, considering those levels, efficient and robust performed settings are found for each parameter by using a fuzzy logic method. Their method is suggested for tuning only algorithms that have a large number of parameters.

Applications of factorial experimental design showed that, although it is useful in the beginning phase of the process, one drawback is that when the number of parameters increases, the required number of runs also exponentially increases [78]. Gunawan et al. [79] suggested a different decomposition method to reduce the parameter space by dividing (can be performed either manually or automatically) the parameters into several disjoint categories. They used Resolution IV Design [80] for the distinction of main effects and interactions of parameters. After decomposition, they applied the configuration method proposed by Gunawan and Lau [78] for each of the parameters. The tuned simulated annealing algorithm with their approach outperforms basic ParamILS and leads to a significant improvement in annual cost savings.

In Table 2, we mentioned the main advantages and disadvantages of irace and CALIBRA methods.

Table 2. A brief summary of advantages and disadvantages of irace and CALIBRA algorithm configuration methods.

Methods	Advantages	Disadvantages
IRACE	• Although irace is primarily developed for solution quality calibration, they added an adaptive capping technique and the resulting version of irace also achieves competitive results for the tuning target of run-time minimization [64]. • The major advantage of this method is coping with multiple instances and with all parameter types [65]. • Furthermore, racing and sharpening ad-hoc methods are used with irace [64,65].	• For a small tuning budget of irace, the obtained configuration may not perform better than a configuration selected randomly [13]. • One important deficiency of the irace is not taking into account instance-configuration correlation and interactions among parameters [7]. • Developed elitist strategy often converges to the good parameter set quickly, and this causes decreases in search of new alternative configurations [7].

Table 2. *Cont.*

Methods	Advantages	Disadvantages
CALIBRA	• Because CALIBRA uses both the experimental designs and local search, it can quickly approximate the promising region [13]. • It is expected that, when parameter values of the tuned algorithm have a major effect on its performance, the results of the CALIBRA process will be more satisfactory [48].	• There are two limitations of CALIBRA, one of it can only tune five parameters, and it is not informative about the interaction effect of parameters, so it can be more efficient when the parameter interactions are negligible [13].

2.4. Model-Based Methods

As we mentioned above, the algorithm configuration problem can be regarded as a specific version of black-box optimization, and so black-box optimization methods can be used for tackling this problem. Model-based optimization methods developed for black-box optimization build a response surface model to model the relation between the algorithm performance and its parameter values and then use this model for the configuration of the target algorithm. Sequential model-based optimization (SMBO) builds a model and uses it for selecting configurations worth further investigation iteratively. The most important difference between model-based tuners from the model-free tuning methods is the usage of response surface methods to determine configurations that will be tested. For details of the SMBO procedure, we refer to [6].

The most notable of the SMBO procedures is the efficient global optimization (EGO) algorithm suggested by [81] that combines the design and analysis of computer experiments (DACE) model, suggested by [82], the expected improvement criterion (see [6]) and a branch and bound method for intensification of the most promising areas. EGO is only capable of dealing with deterministic algorithms or simulations.

There are three extensions of the EGO algorithm. The first of these extensions, the sequential kriging optimization (SKO) algorithm, is suggested by Huang et al. [83]. SKO adds a noise function to the DACE model and accepts the observation noise is normally distributed, which leads to the use of EGO for handling stochastic algorithms. Secondly, Williams et al. [84] suggested another extension that uses a Gaussian process model to optimize marginal performance across a set of "environmental conditions". Their method is expensive and can only be used for optimizing mean performance for a set of the problem instance.

2.4.1. SPO, SPO+, and TB-SPO

The last extension of the EGO algorithm is the sequential parameter optimization (SPO) suggested by Bartz-Beielstein et al. [85]. The SPO procedure starts with defining design point sets (initial configuration) by using Latin Hypercube Sampling (LHS) (available at https://cran.r-project.org/web/packages/lhs/index.html (accessed on 10 June 2022)) such as SKO. To select m design points, the parameter value interval must be divided into m equal intervals, and then a random number is chosen in each interval. The performance of each parameter vector is determined after a number of executions due to the stochasticity of algorithms, and the best-performing point is selected as the initial solution. Then, a stochastic Gaussian model is constructed to represent the relation between the design points and the results and to estimate the algorithm performance. Then new design points that will be used in the proceeding iterations are selected and tested through the intensification mechanism. These new design points consist of the best points found so far and a set of expected good designs (which have the highest expected improvement) according to a model created in previous stages. This model is updated after each iteration based on the best found configuration and is used in the subsequent iteration. The process terminates when the stopping criterion has been reached, and gives one best performing configuration as a result. They evaluated the SPO method in three different cases and

all three scenarios; parameter vectors tuned by SPO increased the performance of the algorithm. They summarized some drawbacks of SPO, such as that it is constrained to decimal and integer values, so it needs the specification of some parameter values.

After researching the key design components of the SPO algorithm, such as choosing the initial design, whether to fit models to raw or log-transformed performance data, the expected improvement criterion, and the intensification criterion, Hutter et al. [86] suggested an improved version called SPO+. In the first part of the study, they compared SPO and SKO, and SPO showed better performance than SKO. For details on the SKO method and implementation differences between the SKO and SPO methods, we refer to [86]. Then they investigated key design components of the SPO method, and, consequently, after evaluating these four, they showed that the log-transformation and the intensification criterion essentially affect its performance. Then they compared the performance of SPO and SPO+ to the ParamILS and demonstrated that SPO+ demonstrated better performance in their experiment.

Bartz-Beielstein et al. [87] used random permutation tests to decide whether a candidate configuration must be tested again in the next iteration as often as the current best configuration or it can be eliminated fairly. Additionally, they gave recommendations for an adequate budget allocation for SPO. Furthermore, their study analyzed the interaction between global and local search in sequential tuning procedures. Lasarczyk [88] applied Chen's *optimal computing budget allocation* (OCBA) [89] to the SPO Toolbox (SPOT). The SPOT package is implemented in the R programming language and it can be achieved on the website [90]. Some example implementations of this package are [91–93].

Hutter et al. [94] suggested the TB-SPO, which considers both the varying amount of time required for different algorithm runs and the complexity of model building and evaluation. TB-SPO is the first model-based optimization procedure for parameter tuning with a user-defined time budget. They used their resulting procedure (dubbed-TB-SPO) for configuring a local search solver, and it was demonstrated that the suggested intensification mechanism reduces time spent on the parameter tuning process and shows better performance than the other state-of-the-art methods. They are planning to improve this method to handle categorical parameters and handle multiple instances in the future.

2.4.2. SMAC

Later, Hutter et al. [6] generalized the components of the time-bounded-SPO procedure to overcome two important restrictions of it. They used a new intensification mechanism; a different response surface model based on the weighted Hamming distance. Additionally, they used random forests for dealing with categorical parameters and multiple instances, as well as a novel selection method for the most promising parameter vector in a large parameter space [6]. They defined two sequential model-based optimization methods for parameter tuning problem; the simple model-free Random Online Adaptive Racing (ROAR) method and the Sequential Model-based Algorithm Configuration (SMAC) method. SMAC starts with an initial configuration set and evaluates its performance on the instance. Then it fits a predictive model (gaussian processes or random forests) of performance based on this information. After that, a new candidate configuration that maximizes an expected positive improvement function is investigated by a multi-start search process. The performance score of this obtained new configuration is determined and this configuration is added to the archive. Subsequently, this process is repeated. The evaluation method of the obtained configuration on the target test problem is similar to that used in FocusedILS, and the incumbent (currently best) configuration is also evaluated on more test problems. The stopping criteria for this evaluation process can be, i.e., run time limit, etc.

They compared the performance of SMAC, ROAR, TB-SPO, GGA, and FOCUSEDILS for different configuration scenarios, aiming to optimize the run time of target algorithms. According to the results of their experiments, SMAC demonstrated a state-of-the-art performance. The proposed SMAC (sequential modeling algorithm configuration) method demonstrated promising performance in a variety of research areas, including continuous black-box

optimization [95,96], machine learning [97], configuring CPLEX solvers for decentralized energy system optimization [98], and tuning inexact solvers parameters [99]. Furthermore, AutoFolio [100] used SMAC for automated tuning of algorithm selection methods.

Hutter et al. [8] further extended SMAC by using two different parallel computing methods to decrease the total amount of time required by the automatic algorithm configuration process. The first one is a multiple independent run of the algorithm. Another method of parallelizing the algorithm is distributing the target algorithm runs over multiple cores, which GGA and FocusedILS inherently use. They showed the effectiveness of parallelization with multiple independent runs for popular algorithm configuration methods, PARAMILS and SMAC. Then they applied fine-grained parallelism to SMAC (called D-SMAC). According to the results of the experiment, D-SMAC outperformed independent parallel runs. For full details about SMAC, we refer to the SMAC user manual [101], and Python (available at https://github.com/automl/SMAC3 (accessed on 10 June 2022) and Java implementations ((http://www.cs.ubc.ca/labs/beta/Projects/SMAC/ (accessed on 10 June 2022)) are available online.

2.4.3. Meta-Tuner

Trindade et al. [102] proposed a model-based tuning method, Metatuner, which is based on the sequential optimization of perturbed regression models. Their experimental results proved that this method finds competitive algorithm configurations when compared with the results of Iterated Racing, SMAC, and ParamILS methods in different problem scenarios and gives information about the interaction between parameters and the relevance of each parameter. However, in the final, they discussed this method's drawbacks, such as the number of parameters that can be tuned, and other issues for advancing and improving their method.

We summarized the main advantages and disadvantages of widely used model-based algorithm configuration methods in Table 3.

Table 3. A brief summary of main advantages and disadvantages of widely used model-based algorithm configuration methods.

Methods	Advantages	Disadvantages
SPO+	• Their intensification mechanism reduces the total required time of the parameter tuning process; and their new prediction models show much better performance than the previous model [14,86].	• Although there are considerable improvements in SMBO-based methods, all of them still have three key restrictions, first one is, that it can handle only numerical parameters; it can only configure the algorithm for a single test problem. Moreover, it has not been an early terminating procedure for poor-performing parameter configurations [6].
SMAC	• SMAC can search the full configuration space which means it does not require discretized parameter space like PARAMILS [6]. • SMAC overcomes two limitations of other SMBO methods, one of them is handling numerical and categorical parameters and can be used for multiple problem instances that show different features [6]. • SMAC considers the instance features and parameter interactions. • Performance prediction model used in SMAC also considers the model's variance/uncertainty in its predictions and this leads to an improvement of the model itself [6,13].	• As a disadvantage, SMAC can only use to optimize the runtime of the algorithm and it cannot deal with all instance types, such as NP-hard problems [10].

3. Comparative Studies

There are some studies conducted to compare the performance of automatic algorithm configuration methods. These comparative studies are scarce in the literature, and Bezerra [103] summarized the reasons as follows: such a representative study requires a set of different benchmark problems and algorithms; it must consider different cases, such as different performance metrics to be optimized; multi- and per-instance setups; benchmarks comprising different parameter types. Additionally, the computational cost of such a study is the most significant difficulty that researchers may face. In this section, we will summarize a few comparative studies of automatic algorithm configuration techniques.

Montero et al. [12] described and compared four automated algorithm configuration procedures: F-Race, Revac, ParamILS, SPO, and the blind random search (BRS) method. Additionally, they analyzed their advantages and disadvantages in terms of quality of results, the effort required, information delivered, usability, user-friendliness, and behavior in different scenarios. A standard genetic algorithm (SGA) used for solving classical continuous test functions was used for the evaluation of these tuning methods. They discussed the requirements of each method, summarized the main global conclusions from their study, and established some guidelines for users that can help to select the most suitable configuration method. For details of the numerical results, we refer to the original paper.

Veček et al. [55] suggested a new algorithm configuration technique that is a combination of a recently developed method for comparing and ranking evolutionary algorithms Chess Rating System for Evolutionary Algorithms (CRS4EAs) and meta-evolution methods. Another important objective of this study is to compare the proposed configuration method with two other procedures: F-Race and Revac. According to the results of this study, all three methods have strengths and weaknesses in different domains (which satisfies the No Free Lunch theorem), and the CRS-Tuning method did not perform worse than the compared techniques.

Črepinšek et al. [104] proposed MOCRS-Tuning (Multiobjective CRS-Tuning), an improved version of CRS tuning, to answer the question of "what must be a sufficient number of problems used in the tuning process to obtain robust enough parameters?" Their results showed that while finding better configurations is not guaranteed when using a bigger subset of problem instances, the best parameter values can also be obtained by using a small problem instance subset. This also essentially reduces the required time for the tuning process.

Smit and Eiben [105] discussed the most crucial issues about configuring evolutionary algorithm parameters, and they executed a modest experimental comparison among selected algorithm configuration methods. Their main goal is to demonstrate the feasibility of using algorithm configuration methods and motivate their usage. They considered three algorithm configuration methods (meta-EA (CMA-ES algorithm is selected as a meta-EA), meta-EDA (Revac), and Sequential Parameter Optimization) and discussed their advantages and disadvantages when configuring evolutionary algorithm parameters. Add-ons, such as racing, sharpening, and a combination of them, are used to increase the search efficiency of tuning methods. For details of these add-on methods used in the algorithm configuration literature, we refer to [105,106]. Revac's performance improved with racing and sharpening, resulting in Revac++. The performance of EA is tested on the Rastrigin function. Although their experiment was conducted on one single test function and one EA, they obtained some interesting conclusions. The most important one is that all of the tested algorithms significantly outperform the human experience-based parameter values. With respect to their conclusion, the rank of the tested three methods is different when regarding different tasks of tuning methods. Finally, based on experimental results, the best method is CMA-ES when the quality of parameter configuration is the primary goal, and SPO when detailed information about the parameter space is required.

In Rasku et al. [107], since there is not a comparative study about automatic algorithm configuration of vehicle routing algorithms, they addressed this problem and compared

algorithm configuration methods, and searched for the most convenient method for configuring vehicle routing algorithms. They compared and evaluated the capabilities and robustness of seven parameter tuning techniques for tuning the vehicle routing metaheuristics. They tuned eight metaheuristics for solving two vehicle routing problem variants, and their performance criteria were the solution quality. In this study, compared parameter tuning techniques are CMA-ES, GGA, I/F-Race, ParamILS, REVAC, SMAC, and URS (uniform random sampling). They demonstrated the improvement obtained by using automatic algorithm configuration methods on vehicle route optimization metaheuristics. The results of their comparison demonstrated that, while some tuning methods perform better in some parameter tuning cases, there is not a unique method that outperforms the others at all times. They also argued how to carefully select and use the right configuration method to improve vehicle routing solver performance. The results of this study can help vehicle routing problem researchers select the most appropriate configuration technique.

Previously mentioned offline configuration techniques require performing a significant number of independent runs of the metaheuristic to obtain meaningful information. Montero et al. [108] investigated the use of this performance information to discard components of a metaheuristic that do not lead to any improvement in its performance, in their terms "ineffective components". They experimentally investigated the information obtained from three parameter tuning methods: ParamILS, F-Race, and Revac. Additionally, experiments showed that ParamILS was the best method for identifying ineffective components. They also summarized the main differences between these three tuning methods. For future work, they suggest studying other configuration methods such as the irace procedure and aiming to discover effective components, not ineffective ones. Araya and Riff [109] suggested an effective model-free fine-tuning method, NODOM-C, for detecting ineffective components/strategies of an optimization algorithm to decrease the target algorithm's complexity without loss of performance (it does not aim to find the best parameter values for a target algorithm configuration scenario like other methods).

We listed the comparative studies of automatic algorithm configuration methods in the Table 4. Because we mentioned some of these studies in the previous sections, we did not refer to them again in this section.

Table 4. List of papers that compare various algorithm configuration methods.

Compared Methods	Tuned Algorithm	Target Problem	Performance Criteria
irace, ParamILS, SMAC [64]	CPLEX, Lingeling, and Spear Solvers	The instance files for these scenarios are from the (AClib) benchmark library.	Algorithm speed
CRS- tuning, F-race, Revac [55]	Artificial Bee Colony (ABC), Differential Evolution (DE)	Benchmark test functions	Max. number of fitness evaluations and number of fitness evaluations they needed to reach optimum
meta-EA, Revac++ and SPO [105]	Evolutionary Algorithms	Rastrigin function	Mean Best Utility
meta-GA, Revac [19]	Genetic Algorithm	Multimodal Landscapes	Mean Best Fitness
GGA++, SMAC [23]	two SAT solver	Industrial SAT instances	Algorithm speed
CALIBRA, ParamILS [47]	three SAT solvers (SAT4J, SAPS, GLS+)	for SAT4J and SAPS various instances from previous SAT competitions; for GLS+, GRID instance set used	Algorithm speed
SMAC, TB-SPO, GGA, ParamILS [6]	SAPS, SPEAR, and CPLEX	sets of instances from various domains	Algorithm speed

Table 4. *Cont.*

Compared Methods	Tuned Algorithm	Target Problem	Performance Criteria
ParamILS, SMAC, D-SMAC [8]	Mixed-integer solver CPLEX	MIP benchmark instances	Optimality gap
TB- SPO, FocusedILS, SPO++ [94]	SAPS (dynamic local search algorithm for (SAT) problem)	quasigroup completion problem (QCP) and the graph-coloring problem	minimize the median number of SAPS search steps required to solve the instance and runtime
F-Race, Revac, ParamILS and SPO [12]	A standard genetic algorithm (SGA)	Benchmark test functions	Amount of evaluations to find the optimum
MetaTuner, Irace, ParamILS and SMAC [102]	Differential Evolution (DE) and (SAPS) algorithm	Benchmark test functions and SAT problem	Mean best fitness for DE and time-to-convergence (for SAPS)
ParamILS, I-Race, and Evoca [110]	SGA, ACOTSP software, and Spear algorithm	Benchmark test functions, Randomly selected Euclidean TSP and SAT-encoded graph coloring problems.	Number of evaluations to the optimum for SGA, Distance to optimum for ACOTSP, and Algorithm speed for Spear

4. Other Studies

As we mentioned before, automatic algorithm configuration is a very wide research area, and there are a considerable amount of studies. Although we classified the exact studies into four main categories, there are some other studies that do not belong to these categories. Additionally, there are some studies that do not propose a new tuning method but consider different aspects of the algorithm configuration problem, such as the effect of transformation of the parameter settings. In this section, we will summarize these studies.

4.1. Other Proposed Algorithm Configuration Methods

In the configuration problems, if all parameters of the target algorithm are only continuous or integer parameters, a continuous black-box optimizer, such as CMA-ES [111], BOBYQA [112], or MADS [113], can be used as a parameter tuning approach when combined with an evaluation method (racing or repeated evaluation).

The MADS (mesh-adaptive direct search) method is suggested by Audet and Orban [113] for the configuration of continuous optimization algorithms, and it is guaranteed to converge to a local optimum of the cost function. Audet et al. [114] extend this method to make it more configurable and introduce the primary Opal framework. They continued to extend this Opal framework in their further studies [115,116].

Yuan et al. [117] evaluated continuous optimization algorithms including CMA-ES, BOBYQA, and MADS algorithms enhanced with uniform random sampling (URS) and the sampling method used in iterated F-Race (IRS). They found that for a few numbers of parameters, BOBYQA works best; however, when the number of parameters is large, CMA-ES is the best working one. After this study, Yuan et al. [118] proposed using a post-selection method. In the first step, a small number of evaluations per configuration are evaluated by the numerical optimizers. In the second step, the most promising parameter values are tested by a racing process. More than one elite configuration can be obtained by this method. The drawbacks and advantages of this method are discussed in [118] and [13].

Pushak and Hoos [119] proposed a novel algorithm configuration method, the golden parameter search algorithm (GPS), which integrates add-ons for algorithm configurators (such as racing or intensification mechanisms) with an improved golden section search [120]. The GPS method optimizes each parameter semi-independently in parallel and has two assumptions: that numeric parameters are uni-modal and that there are no strong interactions between most of the parameters.

4.2. Studies on Different Aspects of the Algorithm Configuration Problem

An important barrier to the improvements in algorithm configuration (AC) is a scarcity of empirical studies and reproducible experiments. The hyper-parameter optimization library HPOlib [121] and the algorithm configuration library AClib [122] was developed to handle this obstacle. However, even with such benchmark libraries available and easy to use to compare different methods, testing new parameter tuning techniques is hard because AC benchmarks cannot be easily set up and the evaluation of the AC method's performance is computationally expensive.

Eggensperger et al. [123] proposed using surrogate benchmarks that consider overall important features of the algorithm configuration cases instead of the expensive original algorithm configuration problem. They generate a new set of eleven surrogate benchmarks, nine of which are for optimizing running time, and the rest are for optimizing solution quality. All procedures are publicly available at (http://www.ml4aad.org/algorithm-analysis/epms/ (accessed on 10 June 2022)). Additionally, these surrogate scenarios can substantially speed up configurators, and they facilitate the comparison and evaluation of the performance of algorithm configuration methods.

Anastacio et al. [124] claimed that default parameter settings of optimization algorithms involve important information because default parameter values are commonly chosen based on deep insights into the algorithm controlled by the parameters, and they can be used in parameter tuning. They compared state-of-the-art configuration methods with and without access to meaningful default parameter vectors to explore the impact of the default configuration on these configuration methods. Their first results showed that widely used automated algorithm configuration methods moderately use the information produced by default parameter settings. Although SMAC rarely benefits from good defaults, default values are very important for irace. Secondly, they studied whether the configuration process could benefit from the information provided in the default by reducing the range of the parameters. They suggested two simple methods, which are guided by default values, for reducing the ranges of the target algorithm's parameters. They improved the performance of automatic configuration methods by decreasing the range of possible values and, thus, the size of the search space. In particular, their reduction techniques increase the efficacy of SMAC for 15 out of the 20 configuration scenarios, that of GGA++ for 7 out of the 9 scenarios, and that of irace for a scenario on which they previously obtained only minor improvements. They also compared their results with the recently suggested warm-starting SMAC [125] and they concluded that default-guided search space reduction provides similar and complementary benefits. Additionally, this method can be implemented in a wider range of automatic configuration methods in contrast to warm-starting.

Eggensperger et al. [126] discussed the pitfalls that have been encountered in algorithm configuration experiments (such as over tuning) and suggested the best ways to discard these pitfalls by developing a tool called GenericWrapper4AC (available at https://github.com/automl/ GenericWrapper4AC (accessed on 10 June 2022)). This package provides an interface between algorithm configuration procedures and target algorithms to obtain reliable, reproducible, and robust algorithm configuration experiments. At the end of their study, they gave general recommendations for effective configuration.

In Franzin et al. [127], the authors studied the effect of transformation of the parameter settings for parameter tuning. They considered five different transformations of a single parameter and examined their effects under various configuration budgets. They demonstrated that it is much more important when the configuration budget is low. Additionally, they also showed how a wrong transformation can be detrimental to the configuration process.

Algorithm configuration methods themselves are based on many heuristics, and this leads to a deficiency of theoretical guarantees. Recently, Kleinberg et al. [128] focused on this problem and developed a general-purpose configuration optimizer called Structured Procrastination, with guarantees of close to the optimal algorithm configuration within a

logarithmic factor of the optimal runtime in a worst-case sense. Moreover, they demonstrated that the gap between the worst-case runtimes of the existing configurators (SMAC, ROAR, ParamILS, GGA, irace) and their solution can be large. The main contribution of their work is that it comes with theoretical guarantees (lower and upper bounds on the runtime), but no empirical illustration is provided. In [129,130], they proposed a simpler method (LEAPSANDBOUNDS) that improves on [128] by finding an approximately optimal configuration with better runtime guarantees on a broader class of problems.

In common, the performance of algorithm configuration methods depends on the definition of parameter search space [131], but Evoca [132] (which is a meta evolutionary algorithm) has demonstrated that it is not sensitive to the parameter's search space definition. Montero and Riff [110] suggested an efficient collaborative approach that associates the tuning process with a parameter search space definition process and so combines Evoca with I-Race and ParamILS.

Developed automated parameter configuration methods (ParamILS, GGA, SMAC, and irace are the most popular ones) have their own parameters that affect their performance, and their default values are set manually by their developers. Dang et al. [133] suggested a meta-tuning process and tested this suggested method on the irace algorithm configuration method. Experimental results confirmed the importance of those parameters and also revealed the complex interactions between them. Results also demonstrated that irace default settings are significantly improved with a confidence level of 99%. In Dang et al. [134], they extended the study of irace on several aspects (such as surrogate models used) and they configured 29 important parameters of the SMAC configurator.

5. Concluding Remarks

Although automatic algorithm configuration techniques have been developed and used for more than a decade, an effective solution to the parameter tuning problem of highly-parameterized algorithms and commercial solvers has recently become possible. This success is thanks to methodological advances in recent configuration procedures. We anticipate that the algorithm configuration topic will be crucial for the development and implementation of metaheuristics and will also be extensively used in industry and academia.

This topic is an abundant research domain, and there are many questions that should be researched and there are open directions for the future, such as:

- An additional effort is essential to reduce the computational cost of the tuning process. Developing different performance evaluation techniques and using efficient sampling methods in the initial phase of the tuning process will facilitate achieving this goal.
- With the fact that different configuration procedures can be successful for various types of algorithm configuration problems, it is still unclear which algorithm configuration method should be chosen for a specific algorithm configuration scenario. Montero et al. [12] suggested developing an automatic selection strategy based on the target algorithm and target problem.
- Additionally, although the open-source codes of these automatic configuration methods are proposed, they require practitioners to have satisfactory programming knowledge. A well-documented and easily-available toolbox, allowing integration of newly developed algorithm configuration techniques, is also an important requirement for end-users and practitioners. There are some toolboxes (such as irace and SPOT) out there, but their feature richness, documentation, and usability could be improved.
- Studies on developing benchmark libraries (such as AClib), which also enable the integration of tuning methods, will help the extension of empirical studies in this area.

Until now, a large collection of independent configuration procedures were suggested and summarized in our paper. However, it is expected that algorithm configuration methods will be improved in a more structured research area [135]. We anticipate that this study will be a guide for understanding the general perspective of the studies in this area.

As a final point, in this paper we provided a short introduction to the framework of the algorithm configuration problem, and also provided a survey about offline algorithm

configuration approaches and related works with these methods. We used a modified classification of tuning methods based on the structure of configuration methods, such as model-free tuning methods, model-based tuning methods, DOE-based tuning methods, and racing methods. After explaining the logic of these methods, we also argued about their main advantages and disadvantages to help researchers or practitioners select the best method for their specific problem. Additionally, then, we mentioned comparative studies of these methods, related research topics with the algorithm configuration problem, and other studies that consider different aspects of this problem. We provided a comprehensive review of automatic algorithm configuration methods for researchers or practitioners. Moreover, some recommendations and possible future directions for this topic are provided as a conclusion.

Author Contributions: Conceptualization, Y.E.; methodology, Y.E.; validation, A.D.; formal analysis, A.D.; investigation, Y.E.; resources, Y.E. and A.D.; writing—original draft preparation, Y.E.; writing—review and editing, Y.E. and A.D.; visualization, Y.E. and A.D.; supervision, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liao, T.; Molina, D.; Stützle, T. Performance evaluation of automatically tuned continuous optimizers on different benchmark sets. *Appl. Soft Comput.* **2015**, *27*, 490–503. [CrossRef]
2. Birattari, M.; Kacprzyk, J. *Tuning Metaheuristics: A Machine Learning Perspective*; Springer: Berlin/Heidelberg, Germany, 2009.
3. Akbaripour, H.; Masehian, E. Efficient and robust parameter tuning for heuristic algorithms. *Int. J. Ind. Eng. Prod. Res.* **2013**, *24*, 143–150.
4. Eiben, A.E.; Smit, S.K. Parameter Tuning for configuring and analyzing evolutionary algorithms. *Swarm. Evol. Comput.* **2011**, *1*, 19–31. [CrossRef]
5. Eiben, A.E.; Smit, S.K. Evolutionary Algorithm Parameters and Methods to Tune Them. In *Autonomous Search*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 15–36.
6. Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Sequential Model-Based Optimization for General Algorithm Configuration. In Proceedings of the International Conference on Learning and Intelligent Optimization, Rome, Italy, 17–21 January 2011; pp. 507–523.
7. López-Ibáñez, M.; Dubois-Lacoste, J.; Cáceres, L.P.; Birattari, M.; Stützle, T. The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **2016**, *3*, 43–58. [CrossRef]
8. Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Parallel Algorithm Configuration. In *Learning and Intelligent Optimization*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 55–70.
9. Ansótegui, C.; Sellmann, M.; Tierney, K. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In *International Conference on Principles and Practice of Constraint Programming*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 142–157.
10. Bezerra, L.C.T.; López-Ibánez, M.; Stützle, T. *Automatic Configuration of Multi-objective Optimizers and Multi-Objective Configuration*; Technical Report TR/IRIDIA/2017-011, IRIDIA; Université Libre de Bruxelles: Brussels, Belgium, 2017.
11. Hoos, H.H. *Automated Algorithm Configuration and Parameter Tuning. Autonomous Search*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 37–71.
12. Montero, E.; Riff, M.C.; Neveu, B. A beginner's guide to tuning methods. *Appl. Soft Comput.* **2014**, *17*, 39–51. [CrossRef]
13. Huang, C.; Li, Y.; Yao, X. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Trans. Evol. Comput.* **2019**, *24*, 201–216. [CrossRef]
14. Schede, E.; Brandt, J.; Tornede, A.; Wever, M.; Bengs, V.; Hüllermeier, E.; Tierney, K. A Survey of Methods for Automated Algorithm Configuration. *arXiv* **2022**, arXiv:2202.01651.
15. Gratch, J.; Dejong, G. *COMPOSER: A Probabilistic Solution to the Utility Problem in Speed-Up Learning*; RIC: Washington, DC, USA, 1992.
16. Minton, S. Automatically configuring constraint satisfaction programs: A case study. *Constraints* **1996**, *1*, 7–43. [CrossRef]
17. Mercer, R.E.; Sampson, J.R. Adaptive search using a reproductive meta-plan. *Kybernetes* **1978**, *7*, 215–228. [CrossRef]
18. Grefenstette, J.J. Optimization of control parameters for genetic algorithms. *IEEE Trans. Syst. Man Cybern.* **1986**, *16*, 122–128. [CrossRef]

19. De Landgraaf, W.A.; Eiben, A.E.; Nannen, V. Parameter Calibration Using Meta-Algorithms. In Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 71–78.
20. Kadioglu, S.; Malitsky, Y.; Sellmann, M.; Tierney, K. ISAC-Instance-Specific Algorithm Configuration. In *ECAI*; IOS Press: Amsterdam, The Netherlands, 2010; Volume 215, pp. 751–756.
21. Ansótegui, C.; Gabas, J.; Malitsky, Y.; Sellmann, M. MaxSAT by improved instance-specific algorithm configuration. *Artif. Intell.* **2016**, *235*, 26–39. [CrossRef]
22. Malitsky, Y.; Mehta, D.; O'Sullivan, B.; Simonis, H. Tuning Parameters of Large Neighborhood Search for the Machine Reassignment Problem. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 176–192.
23. Ansótegui, C.; Malitsky, Y.; Samulowitz, H.; Sellmann, M.; Tierney, K. Model-Based Genetic Algorithms for Algorithm Configuration. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, Buenos Aires, Argentina, 25–31 July 2015; pp. 733–739.
24. Ansótegui, C.; Pon, J.; Sellmann, M.; Tierney, K. PyDGGA: Distributed GGA for Automatic Configuration. In Proceedings of the International Conference on Theory and Applications of Satisfiability Testing, Barcelona, Spain, 5–9 July 2021; pp. 11–20.
25. Nannen, V.; Eiben, A.E. A method for parameter calibration and relevance estimation in evolutionary algorithms. In Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Washington, DC, USA, 8–12 July 2006.
26. Nannen, V.; Eiben, A.E. Efficient Relevance Estimation and Value Calibration of Evolutionary Algorithm Parameters. In Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 103–110.
27. Pelikan, M.; Goldberg, D.E.; Lobo, F.G. A survey of optimization by building and using probabilistic models. *Comput. Optim. Appl.* **2002**, *21*, 5–20. [CrossRef]
28. Smit, S.K.; Eiben, A.E. Using Entropy for Parameter Analysis of Evolutionary Algorithms. In *Experimental Methods for the Analysis of Optimization Algorithms*; Springer: Heidelberg, Germany, 2010; pp. 287–310.
29. Nannen, V.; Smit, S.K.; Eiben, A.E. Costs and Benefits of Tuning Parameters of Evolutionary Algorithms. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Leiden, The Netherlands, 5–9 September 2008; pp. 528–538.
30. Smit, S.K.; Eiben, A.E. Beating the 'World Champion' Evolutionary Algorithm via REVAC Tuning. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
31. Smit, S.K.; Eiben, A.E. Parameter Tuning of Evolutionary Algorithms: Generalist vs. Specialist. In Proceedings of the European Conference on the Applications of Evolutionary Computation, Madrid, Spain, 20–22 April 2010; pp. 542–551.
32. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
33. Smit, S.K. MOBAT. 2009. Available online: http://mobat.sourceforge.net (accessed on 10 June 2022).
34. Zhang, T.; Georgiopoulos, M.; Anagnostopoulos, G.C. S-Race: A multi-Objective Racing Algorithm. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 1565–1572.
35. Zhang, T.; Georgiopoulos, M.; Anagnostopoulos, G.C. SPRINT Multi-Objective Model Racing. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; pp. 1383–1390.
36. Zhang, T.; Georgiopoulos, M.; Anagnostopoulos, G.C. Multi-objective model selection via racing. *IEEE Trans. Cybern.* **2016**, *46*, 1863–1876. [CrossRef]
37. Blot, A.; Hoos, H.H.; Jourdan, L.; Kessaci-Marmion, M.É.; Trautmann, H. MO-ParamILS: A multi-objective automatic algorithm configuration framework. In Proceedings of the International Conference on Learning and Intelligent Optimization, Ischia, Italy, 29 May–1 June 2016; pp. 32–47.
38. Smit, S.K.; Eiben, A.E.; Szlávik, Z. An MOEA-based Method to Tune EA Parameters on Multiple Objective Functions. In Proceedings of the International Conference on Evolutionary Computation Theory and Applications, Valencia, Spain, 24–26 October 2010; pp. 261–268.
39. Dréo, J. Using Performance Fronts for Parameter Setting of Stochastic Metaheuristics. In Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, Montréal, Canada, 8–12 July 2009; pp. 2197–2200.
40. Smit, S.K.; Eiben, A.E. Multi-Problem Parameter Tuning using Bonesa. In *Artificial Evolution*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 222–233.
41. Ugolotti, R.; Cagnoni, S. Analysis of Evolutionary Algorithms Using Multi-Objective Parameter Tuning. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Nanchang, China, 18–20 October 2014; pp. 1343–1350.
42. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
43. Ugolotti, R.; Sani, L.; Cagnoni, S. What Can We Learn from Multi-Objective Meta-Optimization of Evolutionary Algorithms in Continuous Domains? *Mathematics* **2019**, *7*, 232. [CrossRef]
44. Dymond, A.S.; Engelbrecht, A.P.; Kok, S.; Heyns, P.S. Tuning optimization algorithms under multiple objective function evaluation budgets. *IEEE Trans. Evol. Comput.* **2015**, *19*, 341–358. [CrossRef]
45. Dymond, A.S.; Kok, S.; Heyns, P.S. Mota: A many-objective tuning algorithm specialized for tuning under multiple objective function evaluation budgets. *Evol. Comput.* **2017**, *25*, 113–141. [CrossRef]
46. Hutter, F.; Hoos, H.H.; Stützle, T. *Automatic Algorithm Configuration Based on Local Search*; AAAI: Menlo Park, CA, USA, 2007; Volume 7, pp. 1152–1157.

47. Hutter, F.; Hoos, H.H.; Leyton-Brown, K.; Stützle, T. ParamILS: An automatic algorithm configuration framework. *J. Artif. Intell. Res.* **2009**, *36*, 267–306. [CrossRef]
48. Adenso-Diaz, B.; Laguna, M. Fine-tuning of algorithms using fractional experimental designs and local search. *Oper. Res.* **2006**, *54*, 99–114. [CrossRef]
49. Cáceres, L.P.; Stützle, T. Exploring variable neighborhood search for automatic algorithm configuration. *Electron. Notes Discrete Math.* **2017**, *58*, 167–174. [CrossRef]
50. Blot, A.; Pernet, A.; Jourdan, L.; Kessaci-Marmion, M.É.; Hoos, H.H. Automatically Configuring Multi-Objective Local Search using Multi-Objective Optimization. In Proceedings of the International Conference on Evolutionary Multi-Criterion Optimization, Shenzhen, China, 10–13 March 2017; pp. 61–76.
51. Blot, A.; Kessaci, M.É.; Jourdan, L.; Hoos, H.H. Automatic Configuration of Multi-Objective Local Search Algorithms for Permutation Problems. *Evol. Comput.* **2019**, *27*, 147–171. [CrossRef]
52. Blot, A.; Hoos, H.H.; Kessaci, M.É.; Jourdan, L. Automatic Configuration of Bi-Objective Optimization Algorithms: Impact of Correlation Between Objectives. In Proceedings of the IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI), Volos, Greece, 5–7 November 2018; pp. 571–578.
53. Hutter, F.; Babic, D.; Hoos, H.H.; Hu, A.J. Boosting Verification by Automatic Tuning of Decision Procedures. In *Formal Methods in Computer-Aided Design*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 27–34.
54. Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Automated Configuration of Mixed Integer Programming Solvers. In Proceedings of the International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming, Bologna, Italy, 14–18 June 2010; pp. 186–202.
55. Veček, N.; Mernik, M.; Filipič, B.; Črepinšek, M. Parameter tuning with Chess Rating System (CRS-Tuning) for meta-heuristic algorithms. *Inf. Sci.* **2016**, *372*, 446–469. [CrossRef]
56. Hutter, F.; Lindauer, M.; Balint, A.; Bayless, S.; Hoos, H.; Leyton-Brown, K. The configurable SAT solver challenge (CSSC). *Artif. Intell.* **2017**, *243*, 1–25. [CrossRef]
57. Birattari, M.; Stützle, T.; Paquete, L.; Varrentrapp, K. A Racing Algorithm for Configuring Metaheuristics. In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, New York, NY, USA, 9–13 July 2002; pp. 11–18.
58. Maron, O.; Moore, A.W. Hoeffding Races: Accelerating Model Selection Search for Classification and Function Approximation. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 1994; pp. 59–66.
59. Moore, A.W.; Lee, M.S. Efficient Algorithms for Minimizing Cross-Validation Error. In *Machine Learning Proceedings*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 190–198.
60. Conover, W.J. *Practical Nonparametric Statistics*; John Wiley and Sons: Hoboken, NJ, USA, 1999.
61. Balaprakash, P.; Birattari, M.; Stützle, T. Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement. In *International Workshop on Hybrid Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 108–122.
62. Birattari, M.; Yuan, Z.; Balaprakash, P.; Stützle, T. F-Race and Iterated F-Race: An Overview. In *Experimental Methods for the Analysis of Optimization Algorithms*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 311–336.
63. Zhang, B.; Pan, Q.K.; Meng, L.L.; Lu, C.; Mou, J.H.; Li, J.Q. An automatic multi-objective evolutionary algorithm for the hybrid flowshop scheduling problem with consistent sublots. *Knowl.-Based Syst.* **2022**, *238*, 107819. [CrossRef]
64. Cáceres, L.P.; López-Ibáñez, M.; Hoos, H.; Stützle, T. An experimental Study of adaptive capping in irace. In Proceedings of the International Conference on Learning and Intelligent Optimization, Nizhny Novgorod, Russia, 19–21 June 2017; pp. 235–250.
65. Cáceres, L.P.; Bischl, B.; Stützle, T. Evaluating random forest models for irace. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 15–19 July 2017; pp. 1146–1153.
66. Barbosa, E.B.M.; Senne, E.L.F.; Silva, M.B. Improving the performance of metaheuristics: An approach combining response surface methodology and racing algorithms. *Int. J. Eng. Math.* **2015**, *2015*, 167031. [CrossRef]
67. Eriksson, L.; Johansson, E.; Kettaneh-Wold, N.; Wikström, C.; Wold, S. *Design of Experiments*; Principles and Applications, Learn ways AB: Stockholm, Sweden, 2000.
68. Ridge, E. *Design of Experiments for the Tuning of Optimization Algorithms*; Citeseer: Princeton, NJ, USA, 2007.
69. Ridge, E.; Kudenko, D. Sequential Experiment Designs for Screening and Tuning Parameters of Stochastic Heuristics. In Proceedings of the Workshop on Empirical Methods for the Analysis of Algorithms at the Ninth International Conference on Parallel Problem Solving from Nature (PPSN), Reykjavik, Iceland, 9 September 2006; pp. 27–34.
70. Ridge, E.; Kudenko, D. Tuning the Performance of the MMAS Heuristic. In *International Workshop on Engineering Stochastic Local Search Algorithms*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 46–60.
71. Ridge, E.; Kudenko, D. Determining Whether a Problem Characteristic Affects Heuristic Performance. In *Recent Advances in Evolutionary Computation for Combinatorial Optimization*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 21–35.
72. Ridge, E.; Kudenko, D. Tuning an Algorithm Using Design of Experiments. In *Experimental Methods for the Analysis of Optimization Algorithms*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 265–286.
73. Fallahi, M.; Amiri, S.; Yaghini, M. A parameter tuning methodology for metaheuristics based on design of experiments. *Int. J. Eng. Technol. Sci.* **2014**, *2*, 497–521.
74. Park, M.W.; Kim, Y.D. A systematic procedure for setting parameters in simulated annealing algorithms. *Comput. Oper. Res.* **1998**, *25*, 207–217. [CrossRef]

75. Coy, S.P.; Golden, B.L.; Runger, G.C.; Wasil, E.A. Using experimental design to find effective parameter settings for heuristics. *J. Heuristics* **2001**, *7*, 77–97. [CrossRef]
76. Dobslaw, F. A parameter Tuning Framework for Metaheuristics Based on Design of Experiments and Artificial Neural Networks. In Proceedings of the International Conference on Computer Mathematics and Natural Computing, Yantai, China, 10–12 August 2010.
77. Pham, Q.T. Using fuzzy logic to tune an evolutionary algorithm for dynamic optimization of chemical processes. *Comput. Chem. Eng.* **2012**, *37*, 136–142. [CrossRef]
78. Gunawan, A.; Lau, H.C. Fine-Tuning Algorithm Parameters Using the Design of Experiments Approach. In Proceedings of the International Conference on Learning and Intelligent Optimization, Rome, Italy, 7–21 January 2011; pp. 278–292.
79. Gunawan, A.; Lau, H.C.; Wong, E. Real-World Parameter Tuning Using Factorial Design with Parameter Decomposition. In *Advances in Metaheuristics*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 37–59.
80. Montgomery, D.C. *Design and Analysis of Experiments*; John Wiley and Sons: Hoboken, NJ, USA, 2017.
81. Jones, D.R.; Schonlau, M.; Welch, W.J. Efficient global optimization of expensive black-box functions. *J. Glob. Optim.* **1998**, *13*, 455–492. [CrossRef]
82. Sacks, J.; Welch, W.J.; Mitchell, T.J.; Wynn, H.P. Design and analysis of computer experiments. *Stat. Sci.* **1989**, *4*, 409–423. [CrossRef]
83. Huang, D.; Allen, T.T.; Notz, W.I.; Zeng, N. Global optimization of stochastic black-box systems via sequential kriging meta-models. *J. Glob. Optim.* **2006**, *34*, 441–466. [CrossRef]
84. Williams, B.J.; Santner, T.J.; Notz, W.I. Sequential design of computer experiments to minimize integrated response functions. *Stat. Sin.* **2000**, *10*, 1133–1152.
85. Bartz-Beielstein, T.; Lasarczyk, C.W.; Preus, M. Sequential Parameter Optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2–5 September 2005; Volume 1, pp. 773–780.
86. Hutter, F.; Hoos, H.H.; Leyton-Brown, K.; Murphy, K.P. An Experimental Investigation of Model-Based Parameter Optimization: SPO and Beyond. In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, Montreal, QC, Canada, 8–12 July 2009; pp. 271–278.
87. Bartz-Beielstein, T.; Preuss, M. Considerations of Budget Allocation for Sequential Parameter Optimization (SPO). In Proceedings of the Workshop on Empirical Methods for the Analysis of Algorithms, Reykjavik, Iceland, 9 September 2006; pp. 35–40.
88. Lasarczyk, C.W. Genetische Programmierung Einer Algorithmischen Chemie. Ph.D. Thesis, Technische Universität Dortmund, Dortmund, Germany, 2007.
89. Chen, J.; Chen, C.; Kelton, D. Optimal Computing Budget Allocation of Indifference-Zone-Selection Procedures. Working Paper. Available online: http://www.cba.uc.edu/faculty/keltonwd (accessed on 6 January 2005).
90. Available online: https://cran.r-project.org/web/packages/SPOT/index.html (accessed on 10 June 2022).
91. Preuss, M.; Bartz-Beielstein, T. Sequential Parameter Optimization Applied to Self-Adaptation for Binary-Coded Evolutionary Algorithms. In *Parameter Setting in Evolutionary Algorithms*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 91–119.
92. Konen, W.; Koch, P.; Flasch, O.; Bartz-Beielstein, T. *Parameter-Tuned Data Mining: A general Framework*; Cologne University of Applied Sciences: Köln, Germany, 2010; Volume 20.
93. Konen, W.; Koch, P.; Flasch, O.; Bartz-Beielstein, T.; Friese, M. Tuned Data Mining: A benchmark Study on Different Tuners. In Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation, Dublin, Ireland, 12–16 July 2011; pp. 1995–2002.
94. Hutter, F.; Hoos, H.H.; Leyton-Brown, K.; Murphy, K. Time-Bounded Sequential Parameter Optimization. In Proceedings of the International Conference on Learning and Intelligent Optimization, Venice, Italy, 18–22 January 2010; pp. 281–298.
95. Hutter, F.; Hoos, H.; Leyton-Brown, K. An Evaluation of Sequential Model-Based Optimization for Expensive Black-Box Functions. In Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 1209–1216.
96. Tanabe, R.; Fukunaga, A. Tuning Differential Evolution for Cheap, Medium, and Expensive Computational Budgets. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 2018–2025.
97. Thornton, C.; Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 847–855.
98. Schwarz, H.; Kotthoff, L.; Hoos, H.; Fichtner, W.; Bertsch, V. Using Automated Algorithm Configuration to Improve the Optimization of Decentralized Energy Systems Modeled as Large-Scale, Two-Stage Stochastic Programs. In *Working Paper Series in Production and Energy*; Karlsruhe Institute of Technology (KIT): Karlsruhe, Germany, 2017.
99. Mu, Z.; Hoos, H.H.; Stützle, T. The Impact of Automated Algorithm Configuration on the Scaling Behavior of State-Of-The-Art Inexact TSP Solvers. In Proceedings of the International Conference on Learning and Intelligent Optimization, Ischia, Italy, 29 May–1 June 2016; pp. 157–172.
100. Lindauer, M.; Hoos, H.H.; Hutter, F.; Schaub, T. Autofolio: An automatically configured algorithm selector. *J. Artif. Intell. Res.* **2015**, *53*, 745–778. [CrossRef]
101. Hutter, F.; Ramage, S. *Manual for SMAC*; Version v2. 10.03-Master; Department of Computer Science University of British Columbia: Vancouver, BC, Canada, 2015.

102. Trindade, Á.R.; Felipe, C. Tuning metaheuristics by sequential optimization of regression models. *Appl. Soft Comput.* **2019**, *85*, 105829. [CrossRef]
103. Bezerra, L.C. A component-Wise Approach to Multi-Objective Evolutionary Algorithms. Ph.D. Thesis, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, 2016.
104. Črepinšek, M.; Ravber, M.; Mernik, M.; Kosar, T. Tuning Multi-Objective Evolutionary Algorithms on Different Sized Problem Sets. *Mathematics* **2019**, *7*, 824. [CrossRef]
105. Smit, S.K.; Eiben, A.E. Comparing Parameter Tuning Methods for Evolutionary Algorithms. In Proceedings of the IEEE Congress on Evolutionary Computation CEC'09, Trondheim, Norway, 18–21 May 2009; pp. 399–406.
106. Dobslaw, F. Recent Development in Automatic Parameter Tuning for Metaheuristics. In Proceedings of the 19th Annual Conference of Doctoral Students-WDS 2010, Prague, Czech Republic, 1 June 2010.
107. Rasku, J.; Musliu, N.; Kärkkäinen, T. On automatic algorithm configuration of vehicle routing problem solvers. *J. Veh. Routing Algorithms* **2019**, *2*, 1–22. [CrossRef]
108. Montero, E.; Riff, M.C.; Pérez-Caceres, L.; Coello, C.A.C. Are State-of-the-Art Fine-Tuning Algorithms Able to Detect a Dummy Parameter? In Proceedings of the International Conference on Parallel Problem Solving from Nature, Taormina, Italy, 1–5 September 2012; pp. 306–315.
109. Araya, I.; Riff, M.C. A filtering method for algorithm configuration based on consistency techniques. *Knowl. Based Syst.* **2014**, *60*, 73–81. [CrossRef]
110. Montero, E.; Riff, M.C. Effective collaborative strategies to setup tuners. *Soft Comput.* **2020**, *24*, 5019–5041. [CrossRef]
111. Hansen, N.; Ostermeier, A. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.* **2001**, *9*, 159–195. [CrossRef] [PubMed]
112. Powell, M.J. *The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives*; University of Cambridge: Cambridge, UK, 2009; pp. 26–46.
113. Audet, C.; Orban, D. Finding optimal algorithmic parameters using derivative-free optimization. *SIAM J. Optim.* **2006**, *17*, 642–664. [CrossRef]
114. Audet, C.; Dang, C.K.; Orban, D. Algorithmic Parameter Optimization of the DFO Method with the OPAL Framework. In *Software Automatic Tuning*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 255–274.
115. Audet, C.; Dang, C.K.; Orban, D. Efficient use of parallelism in algorithmic parameter optimization applications. *Optim. Lett.* **2013**, *7*, 421–433. [CrossRef]
116. Audet, C.; Dang, K.C.; Orban, D. Optimization of algorithms with OPAL. *Math. Program. Comput.* **2014**, *6*, 233–254. [CrossRef]
117. Yuan, Z.; de Oca, M.A.A.; Birattari, M.; Stützle, T. Continuous optimization algorithms for tuning real and integer parameters of swarm intelligence algorithms. *Swarm Intell.* **2012**, *6*, 49–75. [CrossRef]
118. Yuan, Z.; Stützle, T.; Montes de Oca, M.A.; Lau, H.C.; Birattari, M. An analysis of post-selection in automatic configuration. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 1557–1564.
119. Pushak, Y.; Hoos, H.H. Golden Parameter Search: Exploiting Structure to Quickly Configure Parameters in Parallel. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, Cancun, Mexico, 8–12 July 2020; pp. 245–253.
120. Kiefer, J. Sequential minimax search for a maximum. *Proc. Am. Math. Soc.* **1953**, *4*, 502–506. [CrossRef]
121. Eggensperger, K.; Feurer, M.; Hutter, F.; Bergstra, J.; Snoek, J.; Hoos, H.; Leyton-Brown, K. Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters. In Proceedings of the NIPS workshop on Bayesian Optimization in Theory and Practice, Lake Tahoe, NV, USA, 10 December 2013; Volume 10, p. 3.
122. Hutter, F.; López-Ibánez, M.; Fawcett, C.; Lindauer, M.; Hoos, H.H.; Leyton-Brown, K.; Stützle, T. AClib: A Benchmark Library for Algorithm Configuration. In Proceedings of the International Conference on Learning and Intelligent Optimization, Gainesville, FL, USA, 16–21 February 2014; pp. 36–40.
123. Eggensperger, K.; Lindauer, M.; Hoos, H.H.; Hutter, F.; Leyton-Brown, K. Efficient benchmarking of algorithm configurators via model-based surrogates. *Mach. Learn.* **2018**, *107*, 15–41. [CrossRef]
124. Anastacio, M.; Luo, C.; Hoos, H. Exploitation of Default Parameter Values in Automated Algorithm Configuration. In Proceedings of the Workshop Data Science Meets Optimization, IJCAI, Macao, China, 31 May 2019.
125. Lindauer, M.; Hutter, F. Warmstarting of Model-Based Algorithm Configuration. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
126. Eggensperger, K.; Lindauer, M.; Hutter, F. Pitfalls and best practices in algorithm configuration. *J. Artif. Intell. Res.* **2019**, *64*, 861–893. [CrossRef]
127. Franzin, A.; Cáceres, L.P.; Stützle, T. Effect of transformations of numerical parameters in automatic algorithm configuration. *Optim. Lett.* **2018**, *12*, 1741–1753. [CrossRef]
128. Kleinberg, R.; Leyton-Brown, K.; Lucier, B. Efficiency through Procrastination: Approximately Optimal Algorithm Configuration with Runtime Guarantees. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Melbourne, Australia, 19–25 August 2017.
129. Weisz, G.; György, A.; Szepesvári, C. Leapsandbounds: A Method for Approximately Optimal Algorithm Configuration. *arXiv* **2018**, arXiv:180700755.

130. Weisz, G.; Gyorgy, A.; Szepesvári, C. CAPSANDRUNS: An Improved Method for Approximately Optimal Algorithm Configuration. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 6707–6715.
131. Montero, E.; Riff, M.C.; Rojas-Morales, N. Tuners review: How crucial are set-up values to find effective parameter values? *Eng. Appl. Artif. Intell.* **2018**, *76*, 108–118. [CrossRef]
132. Riff, M.C.; Montero, E. A New Algorithm for Reducing Metaheuristic Design Effort. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC), Cancun, Mexico, 20–23 June 2013; pp. 3283–3290.
133. Dang, N.; Cáceres, L.P.; De Causmaecker, P.; Stützle, T. Configuring Irace using Surrogate Configuration Benchmarks. In Proceedings of the Genetic and Evolutionary Computation Conference, Berlin, Germany, 15–19 July 2017; pp. 243–250.
134. Dang, N.T.T.; Pérez Cáceres, L.; Stützle, T.; De Causmaecker, P. *Configuring Algorithm Parameter Configurators using Surrogate Configuration Benchmarks*; Ku Leuven: Leuven, Belgium, 2017.
135. Hutter, F. Automated Configuration of Algorithms for Solving Hard Computational Problems. Ph.D. Thesis, University of British Columbia, Vancouver, BC, Canada, 2009.