Parameter Optimization Methods Based on Computational Intelligence Techniques in Context of Sustainable Computing

Pankaj Upadhyay and Jitender Kumar Chhabra

Abstract In sustainable computing techniques, we always need to solve lots of optimization problems like design, planning and control, which are extremely hard. Conventional mathematical optimization techniques are computationally difficult. Recent advances in computational intelligence have resulted in an increasing number of nature inspired metaheuristic optimization techniques for effectively solve these complex problems. Mainly, the algorithms which are based on the principle of natural biological evolution and/or collective behavior of swarm have shown a promising performance and are becoming more and more popular nowadays. Most of these algorithms have their some set of parameters. The performance of these algorithm is highly depends on optimal parameter value settings. Prior to running these algorithms, the user must have values of different parameters, such as population size, parameters related to selection, and crossover probability, number of generations etc. That is energy and resource consuming. In this paper we summarize the work in computational intelligence based parameter setting techniques, and discuss related methodological issues. Further we discuss how parameter tuning affects the performance and/or robustness of metaheuristic algorithms and also discusses parameter tuning taxonomy.

Keywords Sustainable computing · Nature inspired metaheuristic · Parameter optimization · Meta-optimization

P. Upadhyay (✉) · J.K. Chhabra
Computer Engineering Department, NIT Kurukshetra, Kurukshetra, Haryana, India
e-mail: pankajupadhyay2006@gmail.com

J.K. Chhabra
e-mail: jitenderchhabra@gmail.com

A.K. Sangaiah et al. (eds.), *Intelligent Decision Support Systems for Sustainable Computing*, Studies in Computational Intelligence 705,
DOI 10.1007/978-3-319-53153-3_6

1 Introduction

Sustainable computing is concerned with computational methods for sustainable economy, environment and society. It is a broad filed which attempts to optimize computational energy and resource utilization using techniques from mathematical optimization and computer science field. For example Intelligent Transportation System (ITS) is used to provide maximum comfort and convenience to its commuters and further minimizes the operating cost, energy consumption and green house emission. It requires lot of optimization process. In sustainable computing we always require to solve lot of optimization problems like design, planning and control, which are computationally hard. In many problems conventional mathematical optimization methods are computationally difficult. Recent advances in computational intelligence have been resulted in an increasing number of nature inspired metaheuristic optimization technique for effectively solve these complex problems [1].

In literature there are various algorithms which are based on nature inspired computing to solve optimization problems, and some of them such as particle swarm optimization (PSO), grey wolf, ant colony optimization (ACO), Artificial bee colony (ABC), genetic algorithm (GA)and gravitational search algorithm (GSA), have been given very prominent results.

Swarm intelligence and nature inspired metaheuristics are most efficient and widely used algorithms for optimization purpose. These algorithms are more efficient over conventional mathematical optimization algorithms [2, 3]. Every optimization algorithm has some strength and some weakness over other optimization algorithms, some work well on certain problem classes while others may not. According to Wolpert and Macready [4], in heuristic search there is no algorithm which gives better results than all other algorithms for all problems. One of the major issues in applying metaheuristics is how to set optimal parameters. Computational complexity is very high to adjust the control parameters of the algorithm to improve its performance on a particular problem. Choosing the optimal parameter values for a single algorithm to solve a single problem is already non-trivial. Parameter setting is an optimization problem itself. Optimal parameter setting is not only affecting the efficiency of actual optimization problem but also improves the performance and robustness of optimization algorithm. All computational intelligence (CI) algorithms are intrinsically dynamic and adaptive process. Hence the uses of fixed parameters that do not change their values during run are against the spirit of dynamism. This implies that performance of algorithm is dependent on whether parameters are static or dynamic in nature. In literature, numerous studies focused on automatic optimal parameter setting. Though various techniques have been proposed in literature, most of them are computationally expensive when the number of parameters is very high.

The objectives of this paper are given as follows:

- To discuss parameter optimization taxonomy.
- To discuss issues and challenges in parameter optimization and why it is so important in computational sustainability.

- To discuss and compare recent computational techniques used for parameter optimization on different performance indicators.

2 Parameter Optimization Taxonomy

Parameter optimization is the process of setting optimal control parameters for optimization technique that is used for optimization problems. Eiben [5, 6], categorizes the adjusting parameters in two categories: parameter tuning and parameter control. The taxonomy for parameter setting is given below in Fig. 1.

- Parameter tuning: Parameter tuning is the approach of finding the best parameter values before starting the algorithm and parameters remain fixed throughout the runtime of the algorithm.
- Parameter control: In parameter control, parameter values are fixed at the beginning but change during execution.
  - Deterministic Parameter Control: This approach is used when algorithm parameter is change by using some deterministic rule. This deterministic rule is used to change the algorithm parameters without using any feedback from the search strategy.
  - Adaptive Parameter Control: This type of parameter control approach is takes place when there is use of feedback to change algorithm parameters.
  - Self-adaptive Parameter Control: In this approach meta-evolution is used to evolve the parameter values during run of baseline algorithm. During evolution better parameter values propagate from one iteration to another.

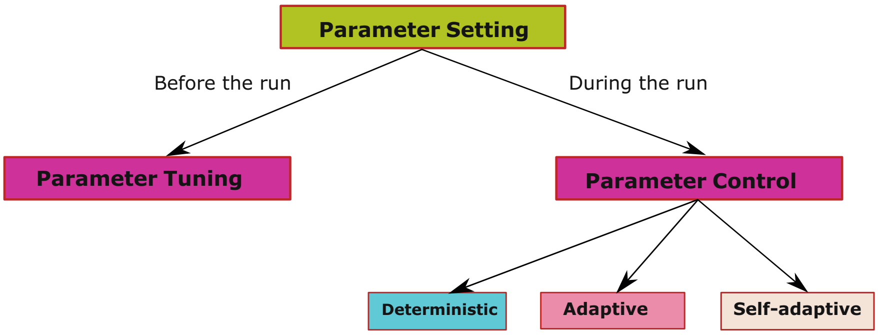


Fig. 1 Taxonomy of parameter setting

3 Parameter Optimization Technique

One of the starting approaches for optimal parameter setting is factorial design, in this technique a comprehensive search is perform to tune the parameter values. This method performs evaluation of the objective function using different parameter values and gets the optimal parameter settings. However, it is time consuming and inefficient so it is often avoided. The cost of such process can be reduced by applying some heuristics, which allow a non-exhaustive search of optimal parameters. Parameter settings are generally chosen in practice by hit and trial method, and tuned by hand [7], taken from other fields, by parameter tuning [8] or by adaptation and self-adaptation mechanisms (parameter control) [9].

In the field of computational intelligence (CI) traditionally there are two main approaches to choose parameter values.

- Parameter tuning, where (good) parameter setting is done before the run of a given CI algorithm. Here, parameter values are remain fixed during CI algorithm is running.
- Parameter control, where (good) parameter setting is fixed during the run of a given CI algorithm. Here, parameter values undergo changes during the run of CI algorithm.

During the last decade there has been extensive research into parameter control. It has been successfully applied in computational intelligence approaches, including Evolution Strategies [10–13], Genetic Algorithms [12, 14], Differential Evolution [15, 16] and Particle Swarm Optimization [17].

For better understanding of underlying parameter tuning approach it is better to further divide it in layered structure. It has three layers namely application layer, algorithm layer and design layer shown in Fig. 2.

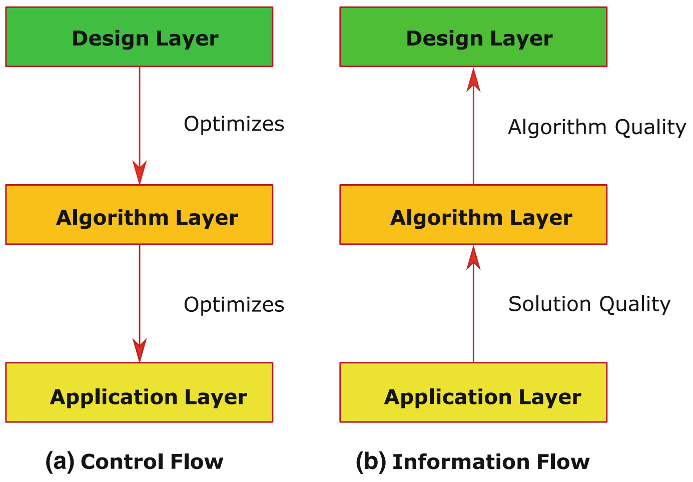


Fig. 2 3 layered architecture of parameter tuning

Table 1 Vocabulary used in parameter optimization

	Problem solving	Parameter tuning
Method	Computational intelligence algorithm	Tuning algorithm
Search space	Solution vector	Parameter vector
Quality	Fitness	Utility
Assessment	Evaluation	Test

The first layer of the architecture contains a design method that is used to find out optimal parameter settings for the underlying computation intelligence technique. This design method can be CI Algorithm or interactive session with user itself. At second layer the algorithm present itself for which parameter setting is to be fixing and at the third layer problem description is available [8]. The vocabulary for distinguishing entities in context of problem solving algorithm and in parameter tuning algorithms is given in Table 1.

The problem solving part contains the underlying CI method to solve the problem where as parameter tuning part contains the metaheuristic method to find optimal parameter setting. The search space for problem is solution vector and for parameter tuning is parameter vector of different values of parameter. Quality check for problem solving method is the fitness function (objective function) and for parameter tuning it is utility.

3.1 Meta-optimization Methods

In the late 1970s by Mercer and Sampson [18] has been given a meta-optimization technique for optimal parameter value settings. It is one of the earliest automatic parameter optimization methods. But due to the large computational costs, their research was very limited. A simple way of finding good behavioral parameters for an optimizer is to employ another overlaying optimizer, called the meta-optimizer. The main concept of meta-optimizer is shown in Fig. 3. In meta-optimization at the lowest level there is optimization problem and at mid level the optimizer to the problem itself. At the top level meta-optimizer is available to solve parameter optimization for problem optimization algorithm.

Another work was done by Grefenstette [19], who also used a GA to optimize the discrete parameters of a GA. He optimized against a set of low dimensional test-function problems to find the generally optimal parameters. In his approach Grefenstette perform extensive experimentation to show the effectiveness of GA as meta-optimizer.

Individuals of population used in meta-optimizer at design layer are parameter vectors of numerical values. Each of the values presented in parameter vector belong to one of the parameter of the underlying CI algorithm to be tuned. To evaluate the utility of parameter vector, the underlying CI algorithm is run several

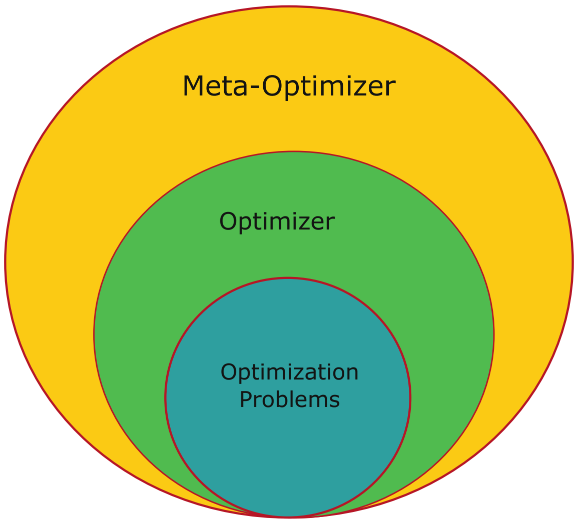


Fig. 3 Meta optimization concept

times using these parameter values. Using this approach for representation and performance (fitness) basically any CI algorithm can be used as meta-optimizer.

Bäck [20], optimize parameter settings by using a specialized hybrid of GAs and ES as meta-optimizer. He used first time a parallel master-slave approach to overcome computational limitations. A more advanced meta-optimization method called REVAC has been developed by Nannen and Eiben [21], which is not only able to optimize the parameters, but also to estimate the relevance of each parameter. Recently Pedersen [22] used the local unimodal sampling (LUS) as a meta-optimizer to find the optimal parameters for a differential evolution (DE) algorithm. Since meta-optimization is very time-consuming, it is not feasible to perform realistic experiments. Now computational power increases manifolds at a scale that allowed to performing realistic experiments.

In literature there are various parameter optimization algorithms based on meta-optimizer. Here are some points about meta-optimization methods.

- In most approaches the implementation is strongly coupled with algorithms that are in use. At meta-level mostly variants of existing metaheuristics were used.
- While specialized meta-level algorithms have the advantage that they can be optimized for a low number of evaluations, the drawback is that they are not easily exchangeable by other existing algorithms.
- A very few authors used parallelization concepts for optimization, but most of them only mentioned that it is highly suitable for meta-optimization.
- Most of the approaches have only aim to optimize the base problem, which turns into poor quality of algorithm. Robustness and performance are considered by only few authors.

3.2 Self Tuning

In [23], Yang et al. have been given a self tuning framework for parameter optimization. In this method algorithm it-self is used to tune the parameters. This kind of parameter tuning is highly expensive and very tough to implement. They demonstrated this framework using firefly optimization algorithm. Proposed algorithm simultaneously tunes parameters itself with actual optimization problem. Parameter tuning and optimality finding has been done simultaneously.

For unconstrained standard optimization problem the goal is to find out global minimum f* of a objective function f(x).

$$\text{Minimize}\, \mathrm{f}(\mathrm{x}),\ \ \text{given}\, \mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \ldots\ldots, \mathrm{x}_\mathrm{d}) \tag{1}$$

An algorithm Algo is used to solve this optimization problem to find the fmin value that is within tolerance to global minimum f*. The aim of parameter tuning is to find out best parameter setting P*. Thus the parameter tuning algorithm can be formulated as follows

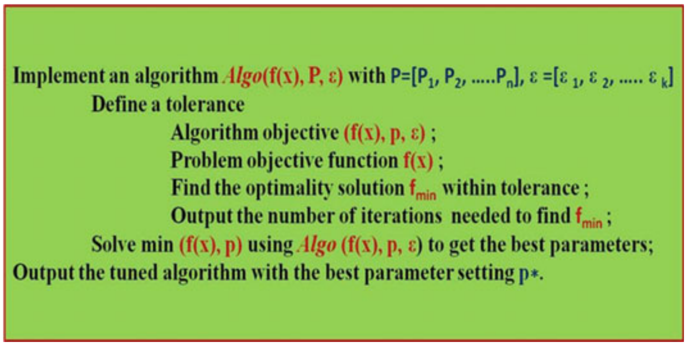
$$\text{Minimize}\ Algo(\mathrm{f}(\mathrm{x}), \mathrm{P}),\ \ \text{given}\, \mathrm{P} = [\mathrm{P}_1, \mathrm{P}_2, \ldots\ldots, \mathrm{P}_\mathrm{n}] \tag{2}$$

In self-tuning framework authors viewed this problem as multi-optimization problem in which two objective functions will be optimized.

$$\text{Minimize}\, \mathrm{f}(\mathrm{x})\ \text{and Minimize}\ Algo(\mathrm{f}(\mathrm{x}), \mathrm{P}) \tag{3}$$

Self tuning algorithm is described as shown in Fig. 4.

There are various approaches to solve multi-objective optimization problem like Pareto optimality, weighted sum. Basically this bi-objective optimization problem can use any of these methods. Here are some concluding remarks about self-tuning parameter optimization.



```
Implement an algorithm Algo(f(x), P, ε) with P=[P_1, P_2, .....P_n], ε =[ε_1, ε_2, ..... ε_k]
    Define a tolerance
        Algorithm objective (f(x), p, ε) ;
        Problem objective function f(x) ;
        Find the optimality solution f_min within tolerance ;
        Output the number of iterations needed to find f_min ;
    Solve min (f(x), p) using Algo (f(x), p, ε) to get the best parameters;
Output the tuned algorithm with the best parameter setting p*.
```

Fig. 4 Self tuning framework

- Optimal parameter setting in any optimization algorithm is highly depends on the very own optimization problem, and there is no unique solution for all problems.
- Self-tuning is very complex in nature and sensitivity analysis is very important for different parameters, since high sensitivity parameters need high degree of tuning.

3.3 Bayesian Case Based Method

Yeguas et al. in [24] proposed a Bayesian case based reasoning method for parameter tuning. The methods discussed in previous sections have the problem of time requirement to evaluate parameters iteratively to get optimize parameter values and interaction of these parameters. This paper evaluates the performance of parameter tuning system empirically to avoid these problems. They combine the Bayesian Networks and Case-Based Reasoning (CBR) method to set optimize parameter values hence maximize the performance of computational intelligence algorithms. In [25, 26] a Bayesian CBR system was introduced as a generic method for solving the parameter tuning problem. The main characteristics which make CBR a good solution for the tuning problem are:

- It is preferable when there is no more information available about the behavioral parameters. This approach uses the past experiences for setting the parameter values.
- It is suitable in problems for which a completely accurate solution does not exist.
- It is suitable when problem instances are very similar in most of the cases.
- The Bayesian CBR system does not require complete information. Furthermore, as the system increases its knowledge, the results improve.

This approach has two important properties, its learning capability, to adapt itself to changes and its capability for autonomy. The design of the Bayesian CBR system consists of mainly two phases. The first phase uses the Bayesian Networks (BNs) relationships among the different parameters. The second phase integrates BNs within a CBR system to solve new problem instances using important features and learning from past for similar problem instances.

3.4 Bi-level Optimization Approach

In early seventies Bracken and McGill [27] introduced bilevel optimization approach in mathematical programming domain. After that numerous studies have been done on bilevel optimization [28, 29]. Most of these bilevel optimization algorithms used nested approach. In nested approach there are two level of optimization, lower level optimization solve the optimization problem for higher level optimization. At lower level Karush-Kuhn-Tucker (KKT) conditions used to transform bilevel optimization into a single level constrained based optimization problem.

In bilevel optimization, there are two nested levels of optimization tasks. Outer one is known as upper level optimization task and inner one is known as lower optimization task. This algorithm has a constraint that lower level optimal solutions can only acceptable as possible input to the upper level. There are mainly two kinds of variable, lower level and upper level. Those are used at lower level optimization task and upper level optimization task respectively. Some characteristics of bilevel parameter optimization are as follows:

- Two nested levels of optimization task.
- Computationally efficient and work well when no. of parameters are high.
- The bilevel optimization converges fast towards optimal parameter settings.

3.5 Additive Procedures

For parameter tuning methods there are some extra additive procedures which increase the capability and search efficiency of parameter tuning methods. These additive procedures are independent of main tuning methods and work as a supplement to main tuner. These additive procedures are given as follows:

- **Racing**: Racing was introduced by Maron and Moore [30]. The racing is used to decrease the number of test cases to determine the quality of parameter vectors, and thus decrease the total runtime of tuning algorithm. The main idea behind racing is that, the number of test cases to determine the utility of a parameter vector is not constant throughout the search. Initially only few test cases are used for each vector and separate out those vector which perform good and increase the number of test cases for those vectors which are not performing worse or better than the good vectors. This approach reduced the significant number of test cases as used in each vector perform tests for each test.
- **Sharpening**: It was introduced by Bartz-Beielstein et al. [31] in Sequential Parameter Optimization (SPO) method. The aim of this sharpening method is also to reduce the number of test cases used to determine the quality of parameter vectors as compared to simple test approach. Initially tuning algorithm start with little number of test cases per vector, as it reaches a certain threshold value, the number of test cases per vector increased to double. Therefore the algorithm explores the search space very fast. This means at the time of termination, the current vector is tested frequently. This leads to very promising results.

4 Algorithm Quality: Performance and Robustness

In computational Intelligence optimization algorithms the accuracy of optimization algorithm is highly dependent on the parameter setting. The performance is highly affected by the optimal parameters setting but parameter optimization is computational overhead and time consuming task. There are several measure to check the performance of computational intelligence techniques some of them are discussed as below.

4.1 Performance Measures

Generally we can check the performance of CI techniques by their solution quality (accuracy) and computational time complexity (speed). The most common performance metrics used in nature inspired computing are as follows:

- Mean of best fitness values over evaluations.
- Average no of evaluations to be performed to get optimal solution.
- Success ratio.

These measures are not always appropriate. If we have a problem which spread of data is very high (large variance), then the algorithm's performance results are questionable. In these cases, mean (and standard deviation) have no significant meaning and it is advisable to use median instead of mean or the best fitness value [32].

Obviously, the actual performance metrics determines the choice of the best parameter settings. Recent studies showed that, different performance measures affect the optimal parameter settings [33]. Without considering the different performance metrics we can't claim anything about optimal parameter settings.

4.2 Robustness

Here robustness means how the performance of optimization algorithm varies over different input parameters. However the performance of any metaheuristic algorithm depends on: problem instance, the parameter vector and the random seed of stochastic process. So there are basically three type of robustness according to problem instance, the parameters and random seed.

If the parameter tuning for an CI algorithm is done using one function over some parameter vector and for some instance of problem do well, it is not necessary that it work well for other problem instances. For robustness to change in parameter values, it is thoroughly measured per parameter individually. These optimization algorithms are stochastic in nature, since they depend on random generations.

So these experimentations require a number of repeated evaluations with identical setup, but with different random seeds.

5 Conclusion and Discussion

This paper provides a comprehensive study of parameter optimization. Accuracy of any computational intelligence (CI) algorithm is highly dependent on optimal settings of parameters. We have discussed how parameter setting affects the performance and robustness of evolutionary algorithms. There is a big question that if we are comparing different parameter optimization algorithm over some benchmark algorithm then whether benchmark algorithm tuned too. Not tuning benchmark algorithm itself is very unreasonable and is biased in nature. So it is equally important to tune benchmark algorithm also tuned before comparison. This paper concludes that parameter optimization depends on various factors and best solution for some problem instance may or may not give optimal solution for other problem instance. Robustness and performance are equally important as optimal parameter settings. In literature there are not much automated tools for parameter optimization so it is open area to develop automated tools for parameter optimization is today's need.

References

1. Y.J. Zheng, S.Y. Chen, Y. Lin, W.L. Wang, Bio-inspired optimization of sustainable energy systems: a review. Math. Probl. Eng. (2013)
2. X.S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications* (Wiley, 2010)
3. A.H. Gandomi, X.S. Yang, A.H. Alavi, Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng. Comput. **29**(1), 17–35 (2013)
4. D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)
5. A.E. Eiben, Z. Michalewicz, M. Schoenauer, J.E. Smith, Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **3**, 124–141 (1999)
6. A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computation*. Natural Computing Series (Springer, 2003)
7. J. Maturana, F. Lardeux, F. Saubion, Autonomous operator management for evolutionary algorithms. J. Heuristics **16**, 881–909 (2010)
8. A.E. Eiben, S.K. Smit, Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**, 19–31 (2011)
9. F. Lobo, C. Lima, Z. Michalewicz, *Parameter Setting in Evolutionary Algorithms*. Studies in Computational Intelligence, vol. 54 (Springer, Heidelberg, 2007)
10. O. Kramer, Evolutionary self-adaptation: a survey of operators and strategy parameters. Evol. Intell. **3**, 51–65 (2010)

11. O.W. Samuel, G.M. Asogbon, A.K. Sangaiah, P. Fang, G. Li, An integrated decision support system based on ANN and Fuzzy_AHP for heart failure risk prediction. Expert Syst. Appl. **68**, 163–172 (2017). Elsevier Publishers
12. A.K. Sangaiah, A.K. Thangavelu, X.Z. Gao, N. Anbazhagan, M.S. Durai, An ANFIS approach for evaluation of team-level service climate in GSD projects using Taguchi-genetic learning algorithm. Appl. Soft Comput. **30**, 628–635 (2015)
13. A.K. Sangaiah, A.K. Thangavelu, An adaptive neuro-fuzzy approach to evaluation of team-level service climate in GSD projects. Neural Comput. Appl. **23**(8) (2013). doi:10.1007/s00521-013-1521-9. Springer Publishers
14. A. Fialho, Adaptive operator selection for optimization. Ph.D. Thesis, Université Paris-Sud XI, Orsay, France (2010)
15. A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization. Trans. Evol. Comput. **13**, 398–417 (2009)
16. R. Mallipeddi, P. Suganthan, Differential Evolution Algorithm With ensemble of Parameters and Mutation and Crossover Strategies, in *Swarm, Evolutionary, and Memetic Computing*. Lecture Notes in Computer Science, vol. 6466 (Springer, Berlin, 2010), pp. 71–78
17. Z.H. Zhan, J. Zhang, Adaptive Particle Swarm Optimization, in *Ant Colony Optimization and Swarm Intelligence*. Lecture Notes in Computer Science, vol. 5217 (Springer, Berlin, 2008), pp. 227–234
18. R.E. Mercer, J.R. Sampson, Adaptive search using a reproductive metaplan. Kybernetes **7**(3), 215–228 (1978)
19. J. Grefenstette, Optimization of control parameters for genetic algorithms. IEEE Trans. Syst. Man Cybern. **16**(1), 122–128 (1986)
20. T. Bäck, *Parallel Optimization of Evolutionary Algorithms*. Lecture Notes in Computer Science, vol. 866 (Springer, Berlin, 1994), pp. 418–427
21. V. Nannen, A. Eiben, A method for parameter calibration and relevance estimation in evolutionary algorithms, in *Genetic and Evolutionary Computation Conference* (2006), pp. 183–190
22. E.M.H. Pedersen, Tuning & Simplifying Heuristical Optimization, PhD thesis, University of Southampton (2010)
23. X.S. Yang, S. Deb, M. Loomes, M. Karamanoglu, A framework for self-tuning optimization algorithms. Neural Comput. Appl. **23**(7–8), 2051–2057 (2013)
24. E. Yeguas, M.V. Luzón, R. Pavónc, R. Lazac, G. Arroyob, F. Díazda, Automatic parameter tuning for evolutionary algorithms using a Bayesian case-based reasoning system. Appl. Soft Comput. **18**, 185–195 (2014)
25. E. Yeguas, R. Joan-Arinyo, M.V. Luzón, Modeling the performance of evolutionary algorithms on the root identification problem: a case study with PBIL and CHC algorithms. Evol. Comput. **19**, 107–135 (2011)
26. R. Joan-Arinyo, M.V. Luzón, E. Yeguas, Parameter tuning of PBIL and CHC evolutionary algorithms applied to solve the root identification problem. Appl. Soft Comput. **11**, 754–767 (2011)
27. J. Bracken, J. McGill, Mathematical programs with optimization problems in the constraints. Oper. Res. **21**, 37–44 (1973)
28. B. Colson, P. Marcotte, G. Savard, An overview of bilevel optimization. Ann. Oper. Res. **153**, 235–256 (2007)
29. S. Dempe, J. Dutta, S. Lohse, Optimality conditions for bilevel programming problems. Optimization **55**(5–6), 505–524 (2006)
30. O. Maron, A. Moore, The racing algorithm: model selection for lazy learners. Artif. Intell. Rev. **11**, 193–225 (1997)
31. T. Bartz-Beielstein, K.E. Parsopoulos, M.N. Vrahatis, Analysis of particle swarm optimization using computational statistics, in *Proceedings of the International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2004)* (2004), pp. 34–37

32. T. Bartz-Beielstein New experimentalism applied to evolutionary computation. Ph.D. Thesis, Universität Dortmund (2005)
33. D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning* (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989)