American Journal of Engineering Research (AJER) 2022
American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-11, Issue-04, pp-26-31
www.ajer.org
Research Paper Open Access

Classification and Tuning Software Parameters

Dušan Tošić
Faculty of Mathematics, University of Belgrade, Studentski trg 16, 11000 Belgrade, Serbia

ABSTRACT: *Today's software is characterized by the use of various parameters. We can classify all parameters into two classes: configuration and optimization. Configuration parameters are used, above all, to adapt the working environment to the user, while optimization should allow to obtain optimal results using certain software. Optimization parameters are more important because these parameters determine what result will be obtained and whether it will be obtained at all. These parameters can be classified in several ways. A classification based on an additional criterion is proposed here - user-friendliness when using the software. It turns out that this classification is significantly related to the manner the parameters are set, i.e. that we can distinguish statically or dynamically set parameters. The paper describes some problems related to parameters depending on the class to which they belong.*
KEYWORDS: *software, parameter, tuning, classification, optimization.*

Date of Submission: 01-04-2022 Date of acceptance: 10-04-2022

I. INTRODUCTION

In recent decades, the architectures of modern computers have become very complex. At the same time, the accompanying software is becoming more complex. Modern software products are expected to have a number of characteristics: reliability, functionality, flexibility, usability, efficiency, reusability, etc. In [19], these software properties are called quality parameters. However, we will define the parameter as follows: it is information expressed by a name, number, or a selected option that is transferred (communicated) to a program by a user or another program. If the parameters are variable, they must be given specific values before using in a program. The set of values assigned to the parameters is called the setting parameters. The setting parameter can significantly affect the functionality of the software. The before mentioned software characteristics (attributes or a properties) are often expressed through one or more parameters. The parameters can be used and set in different ways. All software parameters we can divided into two classes:

- configuration (adaptive) and
- optimizing.

These classes are not disjoint, i.e. some parameter may belong to one and also the other class. Configuration parameters are used to configure the software according to the needs of users, while optimization parameters enable obtaining optimal results using some kind of software.

II. CONFIGURATION PARAMETERS

Configuration parameters are most commonly used to adapt work environments to user needs. Modern operating systems are typical examples of such work environments, but this also includes environments for various processing such as: program creation, word processing, image processing, etc. A simple such environment is described in [21]. In order for such software to be used immediately after installation, its parameters are most assigned default values. The user usually has the ability to change some parameters and adjust the environment to their needs. In fact, commands usually change properties, and each property is expressed through one or more parameters. According to [20], software properties are created at the design, translation or execution stage. One property can have several sub-properties. For example, in Windows 10, during execution, the settings command is used to adjust certain properties by changing the values of the parameters that determine the property. One type of parameters in Windows operating systems are environment variables that can be changed directly by the user. The default values of the configuration parameters are usually set by top experts and usually a smaller number of them need to be changed. We will not consider this type of parameter further.

III. OPTIMIZATION PARAMETERS

The class of optimization parameters consists of the parameters of specific software applications. Some authors only mean this type of parameter by software parameters. These parameters are related to performance, security, robustness and other properties of the software. Setting these parameters is more important than setting the configuration ones because it usually depends on what results will be obtained and whether they will be obtained at all. In [8], concrete examples show how even small changes in parameter values can lead to poor results. Setting optimization parameters is usually done during software execution. However, it can also be performed during software testing and then the initial parameter values are changed. For the problem to be solved, it is important to determine the optimal values of the parameters, i.e. values for which the best solution, to the problem to be solved, will be obtained. Therefore, significant attention of researchers has been paid to finding the optimal configuration of parameters. According to Eiben et al. [10] there are two approaches to setting parameters:

1. parameter tuning or
2. parameter control

These approaches practically introduce two classes of parameters and it can be said that this is the most common classification accepted by many authors [2], [3], [25], [27]. Thus, by using this classification, two classes of parameters are determined: tuned and controlled

The tuning parameter is based on the procedure of finding values for parameters before running the algorithm for given problem. In parameter tuning, once the parameter values are selected, these values remain fixed during the run of the program. In parameter control, a set of initial values are selected and changed during the program run according to some strategies. Software based on parameter tuning consists of two parts: the first part in which the selection of parameters is performed and the second part in which the algorithm for a given problem is implemented. These parameters are, in fact, static set parameters. The second type consists of parameter control, which, after the initial setting, can be changed during the implementation of the algorithm. The parameters selected in this way can be called dynamically set parameters.

However, we will introduce here a more detailed classification of parameters (a somewhat modified classification listed in [7]) considering the convenience of the software for end user, i.e. easy to use. So, we can distinguish the following classes of parameters:

(1) Fixed in advance (pre-fixed)
(2) Pre-adjusted (pre-set)
(3) Dynamic adjustable
(4) Automatic tuned in advance
(5) Automatic and dynamic adjustable

The parameters from classes (1), (2) and (4) belong to the class of statically set parameters, while (3) and (5) belong to the class of dynamically set parameters.

(1) Fixed in advance

This type of parameter is usually represented by constants and their values are set before compiling the program. For example, in a Java program, three fixed parameters can be defined and set as follows:

```
.....
class Apl
{
    final int PAR1 = 55;
    final int PAR2 = 102;
    final double PAR3 = 2.56;
    .....
 }
```

If we want to change the value of a parameter, the program must be recompiled. This way of using parameters is impractical - it can be found in student programs, but almost never in professionally made software.

(2) Pre-adjusted

In this case, the parameters are variable and their values are set during program execution. Usually, when starting the program, the user is first asked to set values of parameters, and then the main part of the application is executed. Problems arise here if the parameters are not set well, the application may not generate results at all. It often happens that practically useless results are obtained. Fig. 1 shows the panel for setting the parameters for the application of the genetic algorithm (GA). We see that there are a dozen parameters that the user needs to set before executing the application. Good knowledge of genetic algorithms is required to set optimal parameter

values. So, expert knowledge for GA is desirable. As a rule, in evolutionary algorithms, especially in genetic ones, a larger number of parameters appear, which stimulated research related to software optimization parameters in general ([25], [2], [4], [23]).

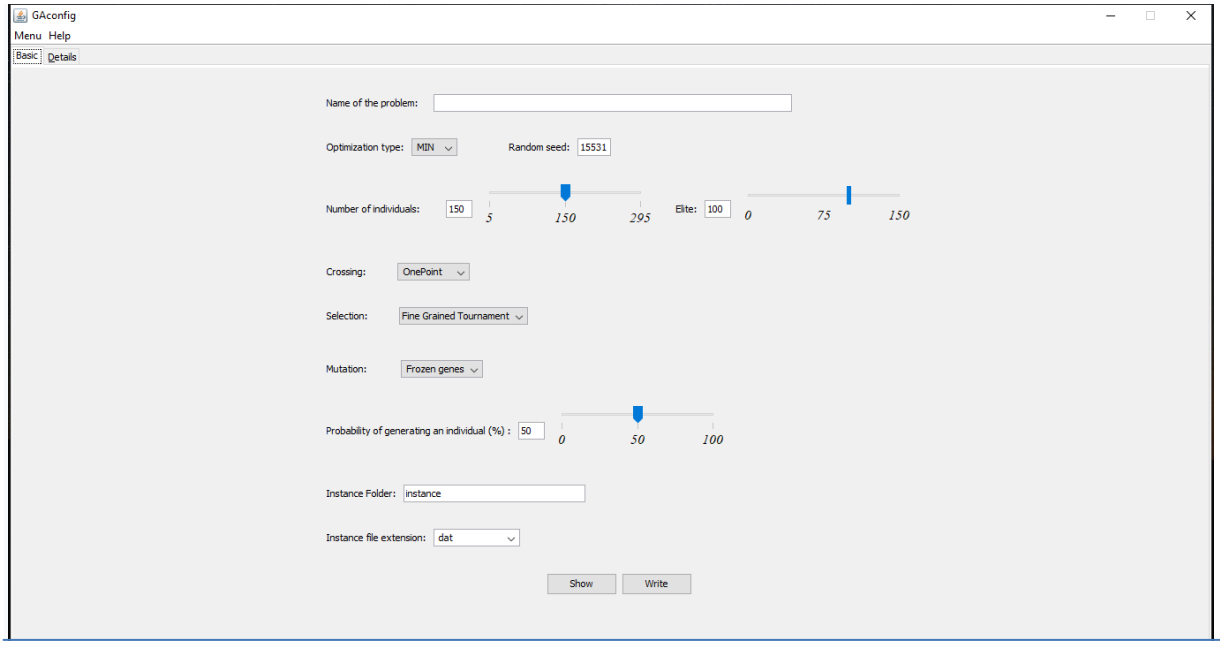

Fig 1. Panel for setting the parameters for the application of the genetic algorithm (GA)

(3) Dynamically adjustable

The characteristic of this type of parameters is that the user can change the initial setting of parameters during the execution of the application. In this case, the help of an expert to select the optimal parameter values is suitable also.

(4) Automatic tuned in advance

Automatic tuned parameters in advance are set before execution of application without user intervention and cannot be reset during application execution. In this case, the help of an expert is not necessary, but the methods of artificial intelligence are used to set the parameters.

(5) Automatic and dynamic adjustable

Automatic and dynamic adjustable parameters are set without user intervention and their values can be changed depending on the program flow. Here, too, the methods of artificial intelligence are used to set parameters. The parameters described in sections 3.1, 3.2 and 3.4 belong to the group of statically set parameters, while the parameters described in 3.3 and 3.5 are dynamically set.

Any kind of parameters should be set so that the software generates optimal results. If such a parameter setting exists, we will call it the optimal setting. A number of questions can be asked about setting parameters. Can we get the optimal setting and meet other software requirements? Which class of parameters to choose for a specific problem? Does the choice of parameters affect the efficiency of the program, etc.? We will consider some of these questions in section 5.

IV. RELATED WORK

A considerable amount of research has been devoted to setting software parameters. Paper [2] and [3] concerned to search-based software engineering (SBSE) techniques for setting parameters. The authors performed empirical analysis on parameter tuning in SBSE analyzing data from more than a million experiments and confirmed that tuning has a critical impact on algorithmic performance. SBSE is a general approach to software engineering in which search based optimization algorithms are used to address different problems in software engineering.

In [6], a method for permanent parameter tuning of optimization algorithms is described. Mathematical formalization of parameter tuning problem and meta-model are described too. The proposed method of permanent parameter tuning was implemented in the automated parameter tuning system allowing: get optimal strategy or set the effectiveness value of the strategy.

The paper [5] contains a description of a new software tool, called Segmentation Parameter Tuning 3 (SPT3).Segmentation is procedure to split the image into discrete meaningful objects and it is used in GEOBIA. SPT3 is designed for automatic tuning of segmentation parameters based on a number of optimization algorithms using different quality metrics as fitness functions.

In [9] a method for setting parameter values based on software testing and machine learning is described. Three Mixed Integer Linear Programming (MILP) solvers with a large number of parameters were considered: GLPK 4.11, CBC 1.01 and CPLEX 9.0. The proposed method (called STOP) allows the values of parameters to be found much better than the default ones using a relatively small number of optimization trials.

[11] and [12] contain a description of the procedure in the integration of existing software into autonomic frameworks. The process takes place in three phases: automating the identification of tuning parameters, rearchitecting to centralize and expose them, and combining these two capabilities to facilitate the integration of existing software into autonomic frameworks. The paper [11] emphasizes the first phase. Parameters (both known and unknown) are identified in the code using static analysis and pattern matching techniques. Software Tuning Panels for Autonomic Control (STAC) project is developed to assist in the transition to more autonomic control. The proposed methodology has been successfully applied to large, open source Java systems.

The paper [13] contains a description of the methodology for setting software parameters based on the use of probabilistic reasoning and decision-making techniques that have been developed by researchers in artificial intelligence, operations research, and other related fields.

In [14], automatic parameter tuning for big data analytics frameworks (BDAF), such as Hadoop MapReduce, Spark, and Dryad, is proposed. These frameworks feature a large number of configuration parameters to users. AutoTune – an automatic parameter tuning system that aims to optimize application execution time on BDAF is presented. Subtle techniques related to big data, related to machine learning, have been used in this system.

The description of the BestConfig system for automatically finding the best configuration setting is presented in [15]. BestConfig (designed with an extensible architecture) uses the divide-and-diverge sampling method and the recursive bound-and-search algorithm to automate the configuration tuning for general systems.

In [16] iTuned, a tool that automates the task of identifying good settings for database configuration parameters, is depicted.

The trade-offs between exact and approximate optimizers for solving a quality-based software selection and hardware mapping problem, from the scalability perspective, is considered in [18].

V. ACHIEVEMENTS AND CHALLENGES RELATED TO PARAMETER SETTING.

The quality of the software significantly depends on the parameters. The existence of parameters contributes to the flexibility of the software. The possibility of determining the optimal parameter setting is especially important. Unfortunately, it is not possible to determine the optimal parameter setting for any problem. It has been mathematically proven in the No Free Lunch (NFL) theorem [24]. In [3], it was explained how NFL theorem refers to algorithms with different parameters. “On average, all algorithms perform equally on all possible problems. For any problem an algorithm is good at solving, there always exist problems for which that algorithm has worse performance than other algorithms. Because the same algorithm with different parameter settings can be considered as a family of different algorithms, the NFL theorem applies to tuning as well. However, the NFL is valid only when all possible search problems are considered.” Although we cannot find the optimal setting of the parameters, the fact is that different solutions of the source problem are obtained for different parameter settings, or are not obtained at all. Is it possible to determine a parameter setting that is as close to optimal as possible? Many researchers have dedicated themselves to solving this problem and have developed various methods for finding "near optimal parameter settings" ([6], [13], [16], [17], [18], [2]).

Various solutions are offered in which, first of all, the parameters described in sections 3.4 and 3.5 are used. In both cases the parameters are set automatically, but different techniques are used to set the parameters. The tuning parameter refers to the parameter from section 3.4, while the control parameter applies to the parameters from section 3.5. The question that arises is: "Which approach is better, parameter tuning or parameter control?" This question is considered in detail in [27] for evolutionary algorithms. The conclusion is: ([27]) ”An obvious answer would be that both have their own strength and weaknesses, so general claims about the superiority of one of the two would be wrong. However, the best static parameter values will never be better than the best scheme that changes them on the fly. Since, in principle, keeping parameter values static is just a special case of parameter control in which the parameter values do not change.” This holds ln general for automatic parameter setting. It can be concluded that the control parameter is a powerful way to set the parameters. However, this way is insufficiently researched or applied in practice. There are three mechanisms for changing parameter values: deterministic, adaptive or self-adaptive ([28]). We will not deal with problems

related to parameter control here, and some of recent results in this area can be found in [27], [28] and [29]. Results of EA research related to parameter control can be applied in other areas.

Unlike parameter control, significantly more results have been obtained by studying parameters tuning. According to [27], depending on the techniques used, there are four approaches to parameters tuning: sampling methods, model-based methods, screening methods, and meta-randomized search algorithms. We will not consider here the various techniques used to adjust the parameters. In order to present some problems that occur with parameter tuning, we will describe in general the most common way of tuning parameters. Adopting the terminology from [17], we denote by *A* target algorithms, i.e. an algorithm with *n* parameters whose performance should be optimized by using a given set of training instances *I*. Different problem instances require different parameter configurations and the goal is to determine an almost optimal setting or each instance. Denote by *B* an algorithm used to tune parameters of *A* and refer to it as the *configurator*. Algorithm *B* defines a *meta software* (meta software is software operating on software) for setting the parameters. Let $p = [p_1, p_2, \ldots, pn]$ be a configuration of parameters where $pi \in [a_i, b_i]$, $\{i=1,n\}$. This determines the parameter configuration space *Q* and we can present it with a Cartesian product $(b_1 - a_1)\ x(b_2 - a_2)x \cdots x(bn - an)$.The space *Q* is huge and evaluating all possible parameter combinations is infeasible in practice. Therefore, various heuristics are used to search for approximately optimal parameter configurations. Denote by *F(p)* a performance metric function which measures the worth of the parameter tuning of algorithm *A* through a series of instances. The automated parameter tuning problem boils down to an optimization problem that seeks $p \epsilon Q$ to minimizes *F(p).* The function *F(p)* is a meta function on *p* (calculated via algorithm *B*) and, as a rule, is very complicated to calculate. For the known algorithm *A*, the execution of algorithm *B* is usually done by choosing a set of training instances *I*, a set of testing instances *T* and determining the configuration $p \epsilon Q$ which minimizes the function *F(p)* to measure the performance of algorithm *A* on a given set *I*. The set of instances *T* is used to execute algorithm *A* using the *p* configuration.

Finding an almost optimal configuration *p* requires extensive calculations ([2], [3], [25]) so that executing meta software can take much longer time than executing target algorithm *A*. Therefore, various methods of artificial intelligence (machine learning, heuristics) are used to memorize certain data and reduce calculations. [1] contains a description the Self-Adapting Numerical Software (SANS) that mediates between the application program and the computational platform. The main component of SANS is the Intelligent Agent (IA) that automates the selection method based on data, target algorithm and system attributes. IA determines parameters using heuristics. To make the choice of parameters to be optimal, computing platform characteristics are also taken into account. Here the parameters are adjusted automatically, but the choice is narrowed only to numerical software. This choice of parameters cannot predict the time required to execute the program because the library could be very rich and it is not known which method will be used.

A new problem arises here: *limited search budget*. The question is: when to stop the search if it takes too long? The time required to execute the program is usually limited. If the problem is such that the search cannot be completed in the available time, instead of automatically setting the parameters, it may be more rational for the user to set the parameters himself as in sections 3.2 and 3.3. It is possible for a certain number of parameters to be fixed by the end user, and for the remaining parameters to be generated automatically ([2], [3]).

VI. CONCLUSION

The use of parameters is inevitable in modern software. The parameters contribute to software flexibility and the comfort of work environments for users. Depending on the type of software, the division of parameters into configuration and optimization is proposed here. Optimization parameters are more significant and there is a large amount of research related to these parameters. These parameters can be further classified and one classification was proposed in Section 3. The main criterion was the ease of use of the software depending on the parameters. If the user has to set the parameter values himself, the software is more inconvenient and vice versa, if the parameter values are generated automatically, the software is more convenient to use. This classification to some extent coincides with the previously introduced division of parameters into tuned and controlled. Problems related to parameter tuning and parameter control have been intensively studied in the previous dozen years and significant results have been obtained, especially for parameter tuning. These results directly relate to the classes of parameters listed in Section 3, which are discussed in more detail in Section 5. Of interest for further study may be the case where the user can fix the values of some parameters, while the values of other parameters are determined automatically.

REFERENCES

[1]. Jack Dongarra, Victor Eijkhout: Self-adapting Numerical Software and Automatic Tuning of Heuristics, ICCS Computational Science — ICCS, 759-767, (2003), DOI 10.1007/3-540-44864-0_78.

[2]. Arcuri, Andrea, and Gordon Fraser. "On parameter tuning in search based software engineering." In: International Symposium on Search Based Software Engineering, Springer, Berlin, Heidelberg, 33-47, (2011).

[3]. Arcuri, A., Fraser, G.: Parameter tuning or default values? An empirical investigation in search-based software engineering. Empir Software Eng 18, 594–623, (2013), https://doi.org/10.1007/s10664-013-9249-9.

[4]. Syafiul Muzid: An Adaptive Approach to Controlling Parameters and Population Size of Evolutionary Algorithm, The 2nd International Conference on Computer Science and Engineering Technology Journal of Physics: Conference Series 1430, 012048 IOP Publishing (2020), doi:10.1088/1742-6596/1430/1/012048

[5]. Diaz, P. M. A., Quirita, V. A. A., Jimenez, L., Garcia, S., Happ, P. N., Feitosa, R.Q., Plaza, A.: A free software tool for automatic tuning of segmentation parameters, In: 5th GEOBIA, Thessalonik, South-Eastern European Journal of Earth Observation and Geomatics, Thessaloniki: Aristotle University of Thessaloniki,v.3,707 – 712, (2014).

[6]. T. Agasiev, A. Karpenko: The program system for automated parameter tuning of optimization algorithms, Procedia Computer Science, vol. 103, 347-354, (2017), https://doi.org/10.1016/j.procs.2017.01.120

[7]. Dušan Tošić: Tuning parameters in complex software systems, 10th Workshop "Software Engineering Education and Reverse Engineering", Ivanjica, Serbia, Sep. (2010), DOI: 10.13140/RG.2.2.26676.40320.

[8]. W. Lee et al., "White-Box Program Tuning", IEEE/ACM International Symposium on Code Generation and Optimization (CGO), Washington, DC, USA, 122-135, (2019), DOI: 10.1109/CGO.2019.8661177.

[9]. Mustafa Baz, J., Paul Brooks, Abhijit Gosavi, Brady Hunsaker: Automated Tuning of Optimization Software Parameters, Technical Report 2007-7, University of Pittsburgh Department of Industrial Engineering Pittsburgh, PA, 1-20, (2007).

[10]. Eiben, Agoston E., Zbigniew Michalewicz, Marc Schoenauer, and James E. Smith. "Parameter control in evolutionary algorithms." In Parameter setting in evolutionary algorithms, Springer, Berlin, Heidelberg, 19-46, (2007).

[11]. Nevon Brake, James R. Cordy, Elizabeth Dancy, Marin Litoiu and Valentina Popescu: Automating Discovery of Software Tuning Parameters, SEAMS '08: Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems, 65–72, (May 2008), https://doi.org/10.1145/1370018.1370031.

[12]. Elizabeth Dancy, James R. Cordy: STAC: Software Tuning Panels For Autonomic Control, Conference: Proceedings of the 2006 conference of the Centre for Advanced Studies on Collaborative Research, October, Toronto, Ontario, Canada, 16-19, (2006), DOI: 10.1145/1188966.1188982.

[13]. David G. Sullivan, Margo Seltzer, Avi Pfeffer: Using probabilistic reasoning to automate software tuning, Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2004, June 10-14, New York, NY, USA, (2004), DOI: 10.1145/1012888.1005739.

[14]. Liang Bao1, Xin Liu2, Weizhao Chen: Learning-based Automatic Parameter Tuning for Big Data Analytics Frameworks, Conference: IEEE International Conference on Big Data (2018), DOI: 10.1109/BigData.2018.8622018.[1

[15]. Yuqing Zhu, Jianxun Liu, Mengying Guo, Yungang Bao, Wenlong Ma, Zhuoyue Liu, Kunpeng Song, Yingchun Yang: BestConfig: Tapping the Performance Potential of Systems via Automatic Configuration Tuning, SoCC '17: Proceedings of the 2017 Symposium on Cloud Computing, 338–350, (September, 2017),https://doi.org/10.1145/3127479.3128605.

[16]. Songyun Duan, Vamsidhar Thummala, Shivnath Babu: Tuning Database Configuration Parameters with iTuned, VLDB '09, Lyon, France, 24-28, (August, 2009), DOI:10.14778/1687627.1687767.

[17]. Lindawati, Hoong Chuin Lau, and David Lo: Instance-Based Parameter Tuning via Search Trajectory Similarity Clustering, In: Coello C.A.C. (eds) Learning and Intelligent Optimization. LION, Lecture Notes in Computer Science, vol 6683. Springer, Berlin, Heidelberg, (2011), DOI https://doi.org/10.1007/978-3-642-25566-3_10

[18]. Dmytro Pukhkaiev, Uwe Aßmann: Parameter Tuning for Self-optimizing Software at Scale, MoST-Rec'19, Beijing, China, (2019).

[19]. Harmeet Kaur, Shahanawaj Ahamad, Gurvinder N. Verma: Identification & Analysis of Parameters for Program Quality Improvement: A Reengineering Perspective, Computer Engineering and Intelligent Systems www.iiste.org ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online) Vol.4, No.5, 23-32, (2013).

[20]. Gargantini, Angelo, Justyna Petke, Marco Radavelli, and Paolo Vavassori. "Validation of constraints among configuration parameters using search-based combinatorial interaction testing." In: International Symposium on Search Based Software Engineering, Springer, Cham, 49-63, (2016).

[21]. Adobe Acrobat SDK 8.1 Parameters for Opening PDF Files for Microsoft Windows, Mac OS, Linux, and UNIX Edition 1.0, April (2007).

[22]. Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E Hassan, and Kenichi Matsumoto: Automated parameter optimization of classification techniques for defect prediction models, In: Proceedings of the 38th International Conference on Software Engineering, 321–332, (2016).

[23]. Agoston E Eiben, Robert Hinterding, and Zbigniew Michalewicz: "Parameter control in ´ evolutionary algorithms", In: IEEE Transactions on evolutionary computation 3.2, 124–141, (1999).

[24]. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1(1), 67–82, (1997).

[25]. Zamani S, Hemmati H: Revisiting hyper-parameter tuning for search-based test data generation, International Symposium on Search Based Software Engineering, 137-152, (2019).

[26]. Selmar K Smit and Agoston E Eiben. "Comparing parameter tuning methods for evolutionary algorithms", In: 2009 IEEE congress on evolutionary computation, IEEE, 399–406, (2009).

[27]. S.K. Smit: Parameter tuning and scientific testing in evolutionary algorithms, Vrije Universiteit, (2012).

[28]. G. Karafotias, M. Hoogendoorn and A. E. Eiben, "Parameter Control in Evolutionary Algorithms: Trends and Challenges," In: IEEE Transactions on Evolutionary Computation, vol. 19, no. 2, 167-187, (April 2015), doi: 10.1109/TEVC.2014.2308294

[29]. Doerr, Benjamin, and Carola Doerr: "Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices",Theory of evolutionary computation, 271-321, (2020).

Dušan Tošić. "Classification and Tuning Software Parameters."*American Journal of Engineering Research (AJER)*, vol. 11(04), 2022, pp. 26-31.