On the Sensitivity of Grid-Based Parameter Adaptation Method

Vasileios A. Tatsis and Konstantinos E. Parsopoulos

Department of Computer Science & Engineering,
University of Ioannina, GR-45110 Ioannina, Greece,
{vtatsis,kostasp}@cse.uoi.gr

Abstract. The rising popularity of metaheuristics has placed their parameter tuning in the center of research in the past decades. It is an important issue with significant implications on their overall performance, especially in demanding problems of high complexity. On the one hand, inappropriate parameters may render the algorithm incapable of detecting good quality solutions. On the other hand, parameter tuning through trial-and-error procedures expands the necessary experimentation time and consumes valuable computational resources. Recently, a general-purpose parameter adaptation method based on grid search in the parameter domain was proposed. The method was successfully demonstrated on two metaheuristics, namely Differential Evolution and Particle Swarm Optimization, attaining competitive performance against other methods. Similarly to other methods, the grid-based search has also a few user-defined parameters. The present work offers a first study of its sensitivity on these parameters. For this purpose, Differential Evolution is the tuned algorithm and the analysis is conducted on the established CEC 2013 test suite. The results verify previous evidence of the method's tolerance on its parameters.

1 Introduction

Metaheuristics have served as efficient solvers for many decades [1]. Among them, evolutionary algorithms have been distinguished for their effectiveness in a plethora of contemporary scientific applications [2, 3]. However, proper parametrization is usually the string attached to their promised success [4]. This deficiency has been tackled through parameter tuning methods, which are distinguished in offline and online methods.

Offline tuning requires a preprocessing phase where the employed metaheuristic is applied on a prescribed set of test problems. Appropriate parameter values are detected through a trial-and-error procedure where a number of different choices are considered and the algorithm's performance is assessed for each one on the test problems. Stochastic search algorithms require the iterative application of this procedure in order to extract statistically sound conclusions. Despite the obvious drawback of the high computational needs, offline approaches offer parameter values that can be reusable in similar problems. On the other hand, failing to select diverse test problems may results in over-specialization of the algorithm. Design of Experiments [5], F-Race [6], and ParamILS [7] are among the state-of-the-art in offline tuning methods.

Contrary to offline tuning, online approaches adapt the parameter setting of the algorithm during its execution, based on its performance during the specific experiment. Although they alleviate the over-specialization problem, they do not offer a single parameter set that can be used in different problem instances or algorithms. On the other hand, their clear advantage is less user intervention in the whole procedure. Concise reviews of online adaptation approaches can be found in [7, 8].

Recently, a general-purpose online parameter adaptation method was proposed in [9]. This method conducts grid search in the parameters' domain during the execution of the algorithm. The search is driven by estimations of the algorithm's neighboring parameter vectors and can be used both for categorical, integer, or real-valued parameters. The method was validated on the Differential Evolution algorithm, which is widely known for its sensitivity on its parameters, as well as on Particle Swarm Optimization [9–11]. The large-scale test problems of the test suite in [12] served as the corresponding testbed. Similarly to other approaches, the method involves a small number of parameters that offer tunability.

The present work constitutes a first study on the sensitivity of the grid-based parameter tuning method on its user-defined parameters. For this purpose, the Differential Evolution algorithm that offered interesting conclusions in previous works is adopted in the present study. Moreover, the established CEC 2013 test suite is considered as the corresponding testbed. Several levels of the basic parameters are considered and their influence on the algorithm's performance is statistically analyzed, offering interesting conclusions.

The rest of the paper is organized as follows: Section 2 briefly presents Differential Evolution, and the grid-based parameter adaptation method is described in Section 3. The experimental configuration for the sensitivity analysis along with the analysis of the results is offered in Section 4. Finally, the paper concludes in Section 5.

2 Differential Evolution

Differential Evolution (DE) [13] is a population-based metaheuristic. After two decades of ongoing development, it is currently considered among the state-of-the-art in evolutionary computation [14]. Given the unconstrained n-dimensional optimization problem

$$\min_{x \in X \subset \mathbb{R}^n} f(x),$$

DE utilizes a population of N search points,

$$P = \{x_1, x_2, \ldots, x_N\}.$$

Each population member is a candidate solution vector

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X, \quad i = 1, 2, \ldots, N,$$

and it is randomly and uniformly initialized in the search space X.

The population is evolved through the iterative application of mutation, crossover, and selection procedures. Mutation consists of the generation of a new vector u_i for each member x_i of the population. The new vector is generated by adding a weighted difference of randomly selected members of the population on a base vector that varies from one mutation operator to another. The baseline mutation operators for x_i are as follows:

$$\text{DE/best/1:} \quad u_i = x_g + F\left(x_{r_1} - x_{r_2}\right), \tag{1}$$

$$\text{DE/rand/1:} \quad u_i = x_{r_1} + F\left(x_{r_2} - x_{r_3}\right), \tag{2}$$

$$\text{DE/current-to-best:} \quad u_i = x_i + F\left(x_g - x_i + x_{r_1} - x_{r_2}\right), \tag{3}$$

$$\text{DE/best/2:} \quad u_i = x_g + F\left(x_{r_1} - x_{r_2} + x_{r_3} - x_{r_4}\right), \tag{4}$$

$$\text{DE/rand/2:} \quad u_i = x_{r_1} + F\left(x_{r_2} - x_{r_3} + x_{r_4} - x_{r_5}\right), \tag{5}$$

where the index g denotes the best member of the population P in terms of function value, i.e.,

$$g = \arg \min_{i=1,\ldots,N} \{f(x_i)\},$$

and $r_1, r_2, \ldots, r_5$, are mutually different, randomly selected numbers between 1 and N that differ also from i. The scale factor F is a user-defined parameter that is crucial for the algorithm since it controls the magnitude of the mutations.

Mutation is followed by crossover, where a trial vector v_i is produced for each x_i by randomly selecting components from the original and the mutated vector. Binomial crossover is defined as follows:

$$v_{ij} = \begin{cases} u_{ij}, & \text{if } \texttt{rand}() \leqslant CR \text{ or } j = R_i, \\ x_{ij}, & \text{otherwise}, \end{cases} \tag{6}$$

where $j \in \{1,2,\ldots,n\}$; `rand()` is a uniform random number generator in the range $[0,1]$; $CR \in (0,1]$ is another user-defined parameter of the algorithm called the crossover rate; and $R_i \in \{1,2,\ldots,n\}$ is a randomly selected index (different for each x_i at each iteration). The crossover procedure is responsible for the amount of information inherited from the original and the mutated vector, while it ensures that at least one component of v_i comes from the mutated vector. An alternative is the exponential crossover, where x_i is initially copied into v_i. Then, a random component of v_i is selected and all subsequent components are replaced by those of the mutated vector u_i until a stochastic condition is satisfied.

The iteration of the algorithm is completed by selection, where each x_i is replaced by the produced trial vector v_i if it achieves better objective function value.The population iteratively evolves until a termination condition is satisfied and the best detected solution x_g is reported.

3 Grid-Based Parameter Adaptation Method

The grid-based parameter adaptation method was initially proposed in [9] and used for the online control of the scalar parameters F and CR of the Differential Evolution algorithm. The method was expanded in [10] for the online adaptation of the mutation operator. The method was further demonstrated for the Particle Swarm Optimization algorithm in [11]. The central idea is the discretization of the parameters' domain and the adaptation of the algorithm's parameter values to neighboring values in the corresponding grid, based on short-run estimations of its performance.

Let us make our description more concrete by considering the Differential Evolution algorithm and its two scalar parameters F and CR, along with two discretization steps λ_{CR} and λ_F for their domains, respectively. Also, let S_{CR}, S_F, be the corresponding discretized sets. Then, the grid is formed as follows:

$$\mathscr{G} = \{(CR, F);\ CR \in S_{CR}, F \in S_F\}.$$

The algorithm starts from a parameter vector in $\mathscr{G}$ (the central point is a reasonable choice), and the population is randomly initialized in the search space of the problem at hand. This population is also called the primary population and it is denoted as P_p. Similarly, its parameter pair is called the primary parameter pair and denoted as (CR_p, F_p). According to the suggestion in [9], P_p is evolved for a number of iterations,

$$t_p = \alpha \times n,$$

where $\alpha > 1$ is an integer and n stands for the problem dimension. In the original work [9], both parameters were considered in the domain $[0,1]$ with $\lambda_{CR} = \lambda_F = 0.1$, and $\alpha = 10$. After the t_p iterations, the primary population stops and the following three phases take place.

Phase I: Cloning

The primary parameter pair (CR_p, F_p) has eight neighboring parameter pairs in $\mathscr{G}$ that are defined as follows [9]:

$$CR' = CR_p + i\,\lambda_{CR}, \quad F' = F_p + j\,\lambda_F, \quad i,j \in \{-1,0,1\}, \tag{7}$$

where the case $i = j = 0$ corresponds to the primary parameter pair itself. For each one of these parameter vectors, a secondary population is defined by cloning the primary population. In [10], the method included also the adaptation of the mutation operator. In this case, four additional secondary populations, also called bridging populations, where defined by cloning the primary population with the primary parameter pair but different mutation operator from the ones defined in Eqs. (1)-(5).

Thus, after the cloning phase we obtain 13 secondary populations denoted as P_{s_j}, $j = 1,2,\ldots,13$, which are identical with the primary one, but 9 of them have the parameter pairs defined in Eq. (7) and same mutation operator with the primary population, while the remaining 4 populations have the primary parameter pair but a different mutation operator each.

Phase II: Performance Estimation

Each one of the 13 secondary populations is individually evolved for t_s iterations in order to reveal its dynamic with the new parameters. Typically, t_s shall be significantly smaller than t_p to spare computational resources (function evaluations). The performance of the secondary populations can be measured using various performance measures. In [9] the average objective value (AOV) of the population was used, which is defined as follows:

$$\mathrm{AOV}_j = \frac{1}{N}\sum_{i=1}^{N} f(x_i), \quad x_i \in P_{s_j}, \quad i = 1,2,\ldots,N, \quad j = 1,2,\ldots,13. \tag{8}$$

This measure reveals the average improvement of the corresponding secondary population with its new parameters. In addition, the objective value standard deviation (OVSD) was also considered in [10], which is defined as follows:

$$\mathrm{OVSD}_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left(f(x_i) - \mathrm{AOV}_j\right)^2}, \quad x_i \in P_{s_j}, \quad i = 1,2,\ldots,N, \quad j = 1,2,\ldots,13. \tag{9}$$

This performance measure determines the diversity of the population's values, which is desirable to alleviate rapid convergence in local minima.

The secondary populations compete using either AOV or both AOV and OVSD (in terms of Pareto dominance when more than one measure is used) and the best one is distinguished. If there are more than one dominant secondary populations, one is selected at random among them.

Phase III: Dynamic's Deployment

The selected secondary population is evolved for t_p iterations to fully reveal its dynamic with the selected parameters. Then, if its AOV improves the AOV of the primary population for at least $\varepsilon \geqslant 0$, the evolved population along with its parameters replaces the primary population. Otherwise, the primary population remains unaltered.

This step completes a full cycle of the method, and the whole procedure is repeated anew from the cloning phase. The maximum number of cycles can be predefined according to the available computational budget [9]. Pseudocode of the method, which is called Differential Evolution with Grid-based Parameter and Operator Adaptation (DEGPOA) is provided in Algorithm 1.

4 Performance Analysis

The DEGPOA algorithm was validated on high- and low-dimensional test suites in [9, 10] with promising results. Without any parameter tuning, the algorithm was capable to compete against other algorithm, controlling its parameters throughout the search procedure. Regarding the grid search parameters, the following values were proposed as default choices [9]:

$$t_s = 5, \quad t_p = 10 \times n, \quad \lambda = \lambda_F = \lambda_{CR} = 0.1, \tag{10}$$

while the initial parameter vector was placed at the center of the grid, i.e., $(CR_p, F_p) = (0.5, 0.5)$, and the initial primary operator was randomly selected from the ones in Eqs. (1)-(5). Thus, the main effect of DEGPOA's parameters remains to be studied.

In the present work we considered three sets of parameter values for t_s, t_p, and λ:

$$S_{t_s} = \{5, 10, 15, 20\}, \quad S_{t_p} = \{5 \times n, 10 \times n, 15 \times n, 20 \times n\}, \quad S_\lambda = \{0.01, 0.05, 0.1, 0.2\}.$$

The previously used version with the parameters of Eq. (10) was considered as the baseline for assessing the new grid search settings. DEGPOA was validated by changing one of its parameters to a different level from the sets above, while keeping the rest of the parameters fixed to the baseline values. This results in 12 new DEGPOA instances.

All experiments were conducted on the established CEC 2013 test suite [15]. This suite consists of 28 unimodal, multimodal, and composite functions, henceforth denoted as f_1-f_{28}. The search space for all test problems is $[-100, 100]^n$, where n stands for the dimension. We considered the

Algorithm 1 DEGPOA: Differential Evolution with Grid-based Parameter and Operator Adaptation

```
1: INITIALIZE($P, F_p, CR_p, O_p$)
2: EVOLVE($P, F_p, CR_p, O_p, t_p$)
3: $m \leftarrow 13$
4: while (NOT TERMINATION) do
5:     /* Phase I: Cloning */
6:     for ($i = 1 : m$) do
7:         if ($i \leqslant 9$) then
8:             /* Secondary population: same operator, different parameters (use Eq. (7)) */
9:             $(P_{s_i}, F_i, CR_i, O_i) \leftarrow (P, F', CR', O_p)$
10:        else
11:            /* Bridging secondary population: different operator, same parameters */
12:            $(P_{s_i}, F_i, CR_i, O_i) \leftarrow (P, F, CR, O'_i)$
13:        end if
14:    end for
15:    /* Phase II: Performance Estimation */
16:    for ($i = 1 : m$) do
17:        EVOLVE($P_{s_i}, F_i, CR_i, O_i, t_s$)
18:    end for
19:    /* Phase III: Dynamic's Deployment */
20:    $(P_{\text{best}}, F_{\text{best}}, CR_{\text{best}}, O_{\text{best}}) \leftarrow$ SELECT_BEST($P_{s_i}, F_i, CR_i, O_i$, AOV, OVSD)
21:    EVOLVE($P_{\text{best}}, F_{\text{best}}, CR_{\text{best}}, O_{\text{best}}, t_p$)
22:    /* Update primary population */
23:    if $\left(\text{AOV}_P - \text{AOV}_{P_{\text{best}}} \geqslant \varepsilon\right)$ then
24:        $(P, F_p, CR_p, O_p) \leftarrow (P_{\text{best}}, F_{\text{best}}, CR_{\text{best}}, O_{\text{best}})$
25:    end if
26: end while
```

most common cases $n = 10$ and $n = 30$ in our study. Also, the guidelines of the test suite dictate that the maximum computational budget is

$$T_{\max} = 10^4 \times n,$$

while the performance is measured by using the error gap between the known optimal solution of the problem, x_{opt}, and the solution x^* achieved by the algorithm,

$$\varepsilon^* = f(x^*) - f\left(x_{\text{opt}}\right).$$

In order to avoid any bias imposed by the initial parameter set, the central parameter $(CR_p, F_p) = (0.5, 0.5)$ was used in all cases. Note that according to the CEC 2013 requirements, a fixed population size $N = 60$ was used and 51 independent experiments were conducted per problem.

Henceforth, we denote as DEGPOA$_{\text{base}}$ the baseline version of the algorithm, and the rest are denoted with corresponding subscripts. For example, the instance with $t_s = 5$, $t_p = 5 \times n$, and $\lambda = 0.01$, is denoted as DEGPOA$_{5s\_5p\_0.01\lambda}$.

Table 1. Comparisons of new DEGPOA instances with DEGPOA$_{\text{base}}$.

	n	W	L	D	W-L	I	NI
t_s modified							
DEGPOA$_{5s\_10p\_0.1\lambda}$	10	0	0	28	0	28	1.00
DEGPOA$_{5s\_10p\_0.1\lambda}$	30	0	0	28	0	28	1.00
DEGPOA$_{10s\_10p\_0.1\lambda}$	10	1	5	22	-4	24	0.86
DEGPOA$_{10s\_10p\_0.1\lambda}$	30	1	2	25	-1	27	0.96
DEGPOA$_{15s\_10p\_0.1\lambda}$	10	0	11	17	-11	17	0.61
DEGPOA$_{15s\_10p\_0.1\lambda}$	30	2	3	23	-1	27	0.96
DEGPOA$_{20s\_10p\_0.1\lambda}$	10	1	12	15	-11	17	0.61
DEGPOA$_{20s\_10p\_0.1\lambda}$	30	4	7	17	-3	25	0.89
		absolute sum:			31		
t_p modified							
DEGPOA$_{5s\_5p\_0.1\lambda}$	10	0	4	24	-4	24	0.86
DEGPOA$_{5s\_5p\_0.1\lambda}$	30	0	2	26	-2	26	0.93
DEGPOA$_{5s\_10p\_0.1\lambda}$	10	0	0	28	0	28	1.00
DEGPOA$_{5s\_10p\_0.1\lambda}$	30	0	0	28	0	28	1.00
DEGPOA$_{5s\_15p\_0.1\lambda}$	10	1	1	26	0	28	1.00
DEGPOA$_{5s\_15p\_0.1\lambda}$	30	1	0	27	1	29	1.04
DEGPOA$_{5s\_20p\_0.1\lambda}$	10	2	1	25	1	29	1.04
DEGPOA$_{5s\_20p\_0.1\lambda}$	30	3	1	24	2	30	1.07
		absolute sum:			10		
λ modified							
DEGPOA$_{5s\_10p\_0.05\lambda}$	10	1	3	24	-2	26	0.93
DEGPOA$_{5s\_10p\_0.05\lambda}$	30	1	2	25	-1	27	0.96
DEGPOA$_{5s\_10p\_0.1\lambda}$	10	0	0	28	0	28	1.00
DEGPOA$_{5s\_10p\_0.1\lambda}$	30	0	0	28	0	28	1.00
DEGPOA$_{5s\_10p\_0.15\lambda}$	10	0	2	26	-2	26	0.93
DEGPOA$_{5s\_10p\_0.15\lambda}$	30	2	0	26	2	30	1.07
DEGPOA$_{5s\_10p\_0.2\lambda}$	10	0	6	22	-6	22	0.79
DEGPOA$_{5s\_10p\_0.2\lambda}$	30	4	7	17	-3	25	0.89
		absolute sum:			16		

The twelve new DEGPOA instances were tested on the CEC 2013 test suite according to the settings above, and their results were recorded and statistically analyzed in order to facilitate comparisons with DEGPOA$_{\text{base}}$. For this purpose, Wilcoxon significance tests at confidence level 95% were used to compare the achieved solution errors. For each comparison of a new instance with the baseline variant, a win was counted if it achieved statistically superior performance than the baseline approach. In the opposite case, a loss was counted, while statistically insignificant differences between algorithms were considered as ties.

Table 1 report the number of wins (denoted as "+"), losses (denoted as "−"), and ties (denoted as "=") of the new DEGPOA instances against DEGPOA$_{\text{base}}$. The fourth column denoted as "W-L" stands for the difference between the number of wins and loses, which provides the general performance trend of the corresponding new instance against the baseline. High positive values correspond to an instance that has far better performance than the baseline, while negative values imply inferior performance of the new instance. The next column denoted as I reports the index value

$$I = 28 + (W - L),$$

which characterizes the relevant performance of the corresponding DEGPOA instance against the baseline over all the 28 test problems. The last column of the table denoted as NI is the normalized index,

$$NI = \frac{\text{Index}}{28},$$

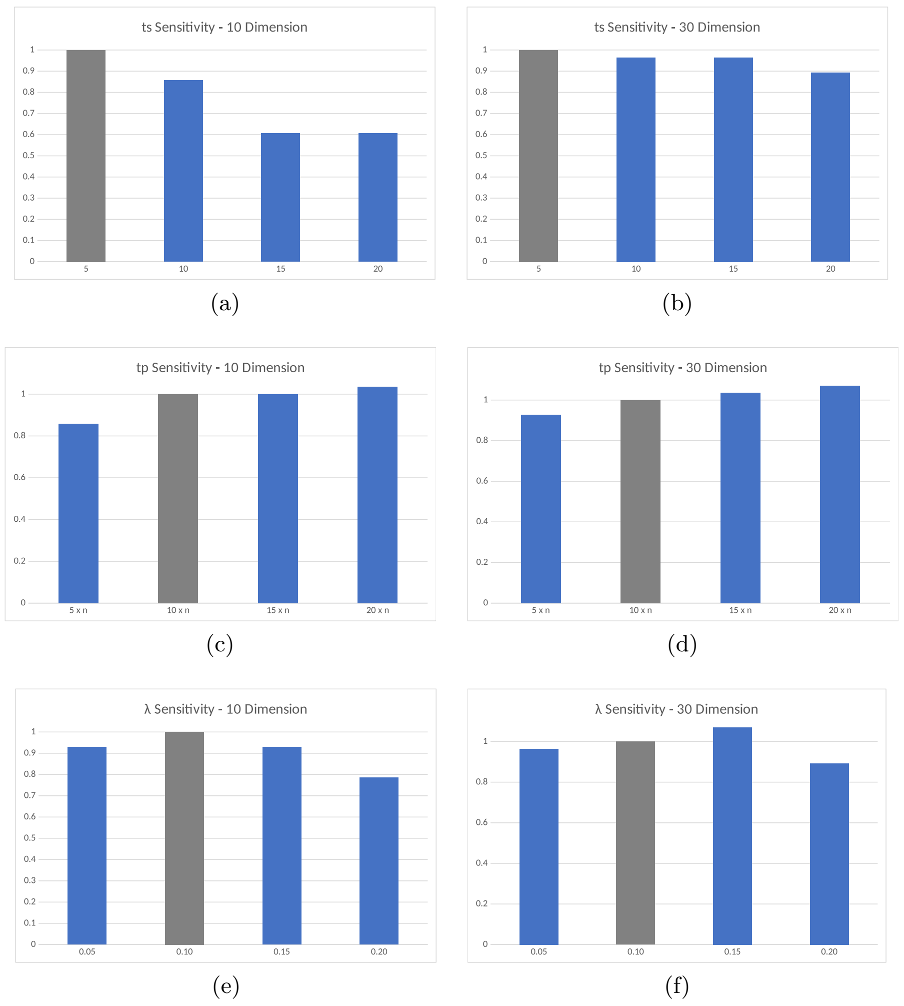

Fig. 1. Values of the normalized index NI per dimension for the parameters t_s (cases (a) and (b)), t_p (cases (c) and (d)), and λ (cases (e) and (f)).

which offers a straightforward comparison measure between the competing algorithms (rounded to 2 decimal digits). Obviously, $NI = 1.00$ when the two compared algorithms have statistically equivalent performance (only ties in statistical tests), while it becomes $NI > 1.00$ whenever the new DEGPOA instance is superior than the baseline, and $NI < 1.00$ when it is inferior. Since $0 \leqslant I \leqslant 56$, the normalized index is bounded in $0 \leqslant NI \leqslant 2$.

In order to facilitate comparisons, Fig. 1 illustrates NI for the different parameter level and dimension. The gray bar stands for the performance of DEGPOA$_{\text{base}}$ while the blue bars refer to the corresponding new instances. The figures offer some interesting conclusions. Firstly, we can observe that t_s can have significant impact on the algorithm's performance in lower dimension ($n = 10$) as we can see in Fig. 1(a). Specifically, smaller values of t_s offer better overall performance, which implies that the estimations of the secondary populations are adequately accurate, sparing computational budget for the dynamic's deployment phase. On the other hand, in the higher-dimensional case ($n = 30$) depicted in Fig. 1(b) this effect becomes milder as a direct consequence of the increased complexity of the problems, which requires longer estimation runs. Nevertheless, the value $t_s = 5$ that was used in previous works [9] verifies its superiority for the specific dimensions.

Regarding the parameter t_p, we can observe in Figs 1(c) and 1(d) that values lower than $10 \times n$ produce inferior performance, implying that the number is inadequate to reveal the primary population and parameters' dynamic. Instead, higher values are beneficial especially for the high-dimensional case. However, the effect remains bounded within 10% of the corresponding baseline value even after doubling the value of t_p. This indicates that the effect of t_p is not highly for the

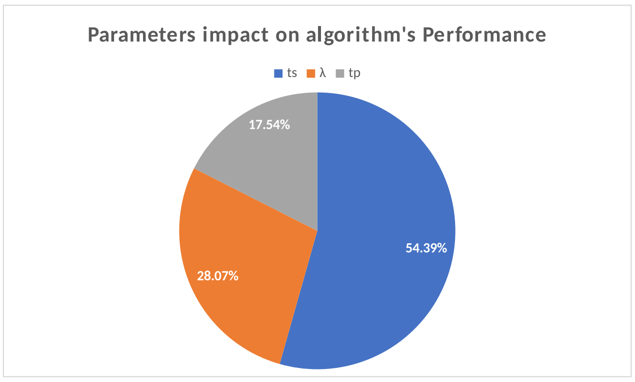

Fig. 2. Overall Parameters impact on algorithm's performance

algorithm's performance if the estimation evaluations t_s retain a proper value. Recall that in all experiments for different t_p values, the default (sub-optimal) $t_s = 5$ value was used.

For both t_s and t_p, the main performance pattern (improving or worsening) was observed for both dimensions. However, this is not the case for the third parameter λ. Changing the discretization step from 0.1 to either lower or higher values produces inferior performance in the 10-dimensional case as illustrated in Fig. 1(e). This motif changes in the higher-dimensional case as illustrated in Fig. 1(f), where slightly increasing λ to 0.15 improves performance for 7%, while different values produce inferior performance of comparable magnitude. Notice that λ determines the search accuracy in the parameter space and has actual dependence both on the algorithm as well as the problem at hand. Thus, there is no clear explanation for this behavior, which is probably the outcome of the interplay between the algorithm's dynamic with the specific parameters and the complexity of the problem itself.

The results show that DEGPOA can achieve stable performance under mild perturbations of the proposed default parameters. In order to identify the overall most influential parameter for all DEGPOA instances, we considered the sum of the absolute differences $W - L$ for each parameter as they are reported in Table 1. Then, we normalized these three values by dividing with their sum, and we received the percentages that are graphically represented in Fig 2. Each normalized value shows the participation of the corresponding parameter in the observed differences. The blue color refers to the t_s parameter, which proves to be the most influential one, followed by λ and t_p.

5 Conclusions

The rising popularity of metaheuristics has placed their parameter tuning in the center of research in the past decades. It is an important issue with significant implications on their overall performance, especially in demanding problems of high complexity. On the one hand, inappropriate parameters may render the algorithm incapable of detecting good quality solutions. On the other hand, parameter tuning through trial-and-error procedures expands the necessary experimentation time and consumes valuable computational resources. Recently, a general-purpose parameter adaptation method based on grid search in the parameter domain was proposed. The method was successfully demonstrated on two metaheuristics, namely Differential Evolution and Particle Swarm Optimization, attaining competitive performance against other methods. Similarly to other methods, the grid-based search has also a few user-defined parameters. The present work offers a first study of its sensitivity on these parameters. For this purpose, Differential Evolution is the tuned algorithm and the analysis is conducted on the established CEC 2013 test suite. The results verify previous evidence of the method's tolerance on its parameters.

Metaheuristics are part of the state-of-the-art in optimization literature for solving demanding problems. However, their performance is strongly dependent on their parameters. This deficiency has resulted in a variety of parameter adaptation methods. Most of them constitute ad-hoc procedures designed for a specific algorithm.

The present work offered a first study of the sensitivity of the recently proposed grid-based parameter adaptation method on the mainstream CEC 2013 test suite. The method was previously validated on the Differential Evolution and Particle Swarm Optimization algorithm for the online control of their parameters. The analysis reveals that the performance estimation phase is the most sensitive one, while the rest of the parameters have only mild influence on the algorithm's dynamic. Also, it reveals that the previously proposed default parameters are very efficient.

Future work will expand the analysis in order to reveal possible interactions between the parameters by using methodologies such as the analysis of variance.

References

1. Gendreau, M., Potvin, J.: Handbook of Metaheuristics, 2nd edn. Springer New York Dordrecht, Heidelberg London (2010)
2. Gogna, A., Tayal, A.: Metaheuristics: review and application. Journal of Experimental & Theoretical Artificial Intelligence 25 (2013) 503–526
3. Torres-Jiménez, J., Pavón, J.: Applications of metaheuristics in real-life problems. Progress in Artificial Intelligence 2 (2014) 175–176
4. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3 (1999) 124–141
5. Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation: The New Experimentalism. Springer (2006)
6. Birattari, M.: Tuning Metaheuristics: A Machine Learning Perspective. Springer (2009)
7. Hoos, H.H.: Automated algorithm configuration and parameter tuning. In Hamadi, Y., Monfroy, E., Saubion, F., eds.: Autonomous Search. Springer, Berlin Heidelberg (2011) 37–72
8. Eiben, A.E., Smit, S.K.: Evolutionary algorithm parameters and methods to tune them. In Hamadi, Y., Monfroy, E., Saubion, F., eds.: Autonomous Search. Springer, Berlin Heidelberg (2011) 15–36
9. Tatsis, V.A., Parsopoulos, K.E.: Differential evolution with grid-based parameter adaptation. Soft Computing 21 (2017) 2105–2127
10. Tatsis, V.A., Parsopoulos, K.E.: Grid search for operator and parameter control in differential evolution. In: Proceedings of the 9th Hellenic Conference on Artificial Intelligence. SETN '16, New York, NY, USA, ACM (2016) 7:1–7:9
11. Tatsis, V.A., Parsopoulos, K.E.: Grid-based parameter adaptation in particle swarm optimization. In: Proceedings of the 12th Metaheuristics International Conference (MIC 2017). (2017)
12. Lozano, M., Herrera, F., Molina, D.: Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. Soft Computing 15 (2011) 2085–2087
13. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optimization 11 (1997) 341–359
14. Das, S., Suganthan, P.N.: Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15 (2011) 4–31
15. : Complementary material: Special session & competition on real-parameter single objective optimization at cec-2013. (`http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm`)