Chapter 22
Experimental Sensitivity Analysis of Grid-Based Parameter Adaptation Method

Vasileios A. Tatsis and Konstantinos E. Parsopoulos

Abstract Grid-based parameter adaptation method has been recently proposed as a general-purpose approach for online parameter adaptation in metaheuristics. The method is independent of the specific algorithm technicalities. It operates directly in the parameter domain, which is properly discretized forming multiple grids. Short runs of the algorithm are conducted to estimate its behavior under different parameter configurations. Thus, it differs from relevant methods that usually incorporate ad hoc procedures designed for specific metaheuristics. The method has been demonstrated on two popular population-based metaheuristics with promising results. Similarly to other parameter tuning and control methods, the grid-based approach has three decision parameters that control granularity of the grids and length of algorithm runs. The present study extends a preliminary analysis on the impact of each parameter, based on experimental statistical analysis. The differential evolution algorithm is used as the targeted metaheuristic, and the established CEC 2013 test suite offers the experimental testbed. The obtained results and analysis verify previous evidence on the method's parameter tolerance, offering also an insightful view on the parameters interplay.

22.1 Introduction

Metaheuristics have been long established as essential search and optimization procedures that can offer (sub-) optimal solutions in cases where traditional optimization methods are either not applicable or deficient [8, 9, 21]. All metaheuristics typically have a number of parameters that influence their dynamic. For example, the popular

V. A. Tatsis (✉) · K. E. Parsopoulos
Department of Computer Science and Engineering, University of Ioannina,
45110 Ioannina, Greece
e-mail: vtatsis@cse.uoi.gr

K. E. Parsopoulos
e-mail: kostasp@cse.uoi.gr

F. Yalaoui et al. (eds.), *Heuristics for Optimization and Learning*,
Studies in Computational Intelligence 906,
https://doi.org/10.1007/978-3-030-58930-1_22

evolutionary algorithms require from the user to set parameters such as population size, mutation and crossover type and rate, among other [6]. The ongoing research activity on the development of new metaheuristics or improvement of established ones has kept the parameter setting problem in a salient position among the most hot research topics in the relevant literature.

There are two main types of parameter setting methodologies in metaheuristics: offline parameter tuning and online parameter control. While offline methods are based on preliminary trial-and-error experimentation, online methods dynamically adapt the parameters on the fly. Offline methods are better suited to cases where a multitude of related problems need to be solved. In such cases, a training subset of the problems can be used to identify a promising parameter setting that is then adopted for solving problems of the same type. Obviously this approach adds in reusability of the parameters but the computational cost of the preliminary experiments can be prohibitively high. Also, over-specialization is a deficiency that shall be taken into consideration. Design of experiments [2], F-race [3], sequential model-based optimization [11], and paramILS [10] are typical examples of such methods.

Alternatively, online parameter control does not provide reusable results, but it dynamically controls the parameters based on performance feedback received during the algorithm run. Thus, the algorithm can adjust its behavior in order to maintain appropriate trade-off between exploration and exploitation during the different phases of the optimization procedure. For this purpose, ad hoc methodologies developed for the specific algorithm are usually adopted. Various online adaptation approaches designed for the differential evolution algorithm that will be later used can be found in [4, 6, 7, 13, 15, 22].

The general-purpose grid-based parameter adaptation method (henceforth abbreviated as GPAM) was recently proposed in [17]. It belongs to the category of online parameter adaptation methods, and it has been successfully applied on two popular metaheuristics, namely differential evolution and particle swarm optimization. It is based on grid search in a discretized parameter domain, exploiting performance estimations through short runs of the algorithm under alternative parameter settings. GPAM offered very promising results in state-of-the-art testbeds of various dimensions [16–19].

All parameter adaptation methods involve a number of inner decision parameters. The number of these parameters shall be reasonably small otherwise the adaptation method would require another external procedure to "tune the tuner". Besides the small number of inner decision parameters, mild parameter sensitivity is highly desirable for adaptation methods. Thus, identifying the impact of each parameter as well as possible interactions among them is an important issue for the effectiveness of the method.

A first study on the sensitivity of GPAM on its parameters was offered in [20]. In that study the main effect of various levels of the three parameters, individually, was investigated by changing one parameter at a time. The results offered some interesting initial insight on the most important parameter. The present work aims at extending the previous study through extensive experimentation and statistical analysis, in order to identify also the interplay between the parameters. For this purpose, differential

evolution was selected as the underlying algorithm, and the state-of-the-art CEC 2013 test suite was adopted as our testbest. Several levels of the GPAM parameters were considered and full factorial experimentation was conducted, accompanied by hypothesis testing analysis.

The rest of the chapter is organized as follows: Sect. 22.2 offers brief presentation of GPAM and differential evolution. The experimental configuration and analysis is offered in Sect. 22.3. Finally, Sect. 22.4 concludes the chapter.

22.2 Background Information

In the following paragraphs, the differential evolution algorithm is briefly presented along with the GPAM approach.

22.2.1 Differential Evolution

Differential evolution (henceforth abbreviated as DE) [14] is a state-of-the-art metaheuristic for numerical optimization problems. Its adaptability and simplicity has placed it among the most popular metaheuristics [5], despite its known sensitivity on its control parameters. For the general continuous bound-constrained n-dimensional optimization problem:

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

DE utilizes a population of N search points:

$$P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\},$$

with $\mathbf{x}_i \in X$, for all $i \in I \stackrel{\text{def}}{=} \{1, 2, \ldots, N\}$. All population members are randomly initialized in the search domain X.

At each iteration t, the mutation, crossover, and selection operators are applied on each $\mathbf{x}_i$. Mutation produces a new vector $\mathbf{u}_i$ for each $\mathbf{x}_i$ by combining randomly selected members of the population. The most common mutation operators of one difference vector are defined as:

$$(\text{DE}/\psi/1) \qquad \mathbf{u}_i^{(t+1)} = \mathbf{x}_{\alpha_1}^{(t)} + F\left(\mathbf{x}_{\alpha_2}^{(t)} - \mathbf{x}_{\alpha_3}^{(t)}\right),$$

where $F > 0$ is a scalar parameter. For $\psi =$ "best" we set $\alpha_1 = g$, and $\alpha_2, \alpha_3 =$ `randi`(I), where g stands for the index of the best member of the population (i.e., the one with the smallest function value), while each call at `randi`(I) returns a random integer from the indices set I defined above. Alternatively, for $\psi =$ "rand"

we set $\alpha_1, \alpha_2, \alpha_3 = \texttt{randi}(I)$. In both cases above, it shall hold that $\alpha_j \neq \alpha_k \neq i$, for all $j \neq k$. Alternative operators with two difference vectors are defined as:

$$\text{(DE / } \psi \text{ / 2)} \qquad \mathbf{u}_i^{(t+1)} = \mathbf{x}_{\alpha_1}^{(t)} + F\left(\mathbf{x}_{\alpha_2}^{(t)} - \mathbf{x}_{\alpha_3}^{(t)} + \mathbf{x}_{\alpha_4}^{(t)} - \mathbf{x}_{\alpha_5}^{(t)}\right)$$

where for $\psi =$ "best" we set $\alpha_1 = g$, and $\alpha_2, \alpha_3, \alpha_4, \alpha_5 = \texttt{randi}(I)$, while for $\psi =$ "rand" we have $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5 = \texttt{randi}(I)$. Another variant with $\psi =$ "current-to-best" defines $\alpha_1 = i, \alpha_2 = g, \alpha_3 = i$, and $\alpha_4, \alpha_5 = \texttt{randi}(I)$. Again, $\alpha_j \neq \alpha_k \neq i$ shall hold for all $j \neq k$. The above cases define the five most common mutation operators.

Mutation is succeeded by crossover, where a trial vector $\mathbf{v}_i$ is produced for each $\mathbf{x}_i$. Each component of $\mathbf{v}_i$ is selected as the corresponding component of $\mathbf{u}_i$ with probability $CR \in (0, 1]$ (another scalar parameter of the algorithm), otherwise it is taken from $\mathbf{x}_i$:

$$\mathbf{v}_{ij} = \begin{cases} \mathbf{u}_{ij}, \text{ if } (\texttt{rand}() \leqslant CR) \texttt{ OR } (j = \zeta), \\ \mathbf{x}_{ij}, \text{ otherwise.} \end{cases}, \qquad j \in \{1, 2, \ldots, n\},$$

where $\zeta = \texttt{randi}(1, 2, \ldots, n)$ is a randomly selected dimension component that is automatically inherited from the mutated vector, and $\texttt{rand}()$ is the pseudo-random number generator in the range [0, 1]. Eventually, the new vector $\mathbf{v}_i$ competes with $\mathbf{x}_i$ and, if it has a better function value, it replaces $\mathbf{x}_i$ in the population for the next iteration. Detailed presentation of the DE algorithm can be found in relevant sources such as [12].

22.2.2 Grid-Based Parameter Adaptation Method

Let us now describe the GPAM proposed in [17] on the DE algorithm. The two scalar parameters of DE are F and CR as presented in the previous section. The mutation operator type can be considered as a categorical parameter. GPAM can tackle such parameters as it was shown in [16].

Given discretization steps λ_{CR} and λ_F for the corresponding scalar parameters, we define their discrete domains S_{CR}, S_F, respectively. Then, we can define a parameter grid as [17]:

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

Smaller step sizes define fine-grained grids that may require longer grid searches for the detection of appropriate parameter pairs. In general, optimal step sizes depend on the studied algorithm and the optimization problem at hand, as well as on the available computational resources. Possible previous experience on the algorithm's parameter sensitivity may provide useful insight for the proper selection of step sizes.

GPAM starts with an initial parameter pair in $\mathcal{G}$ (the central point is a reasonable choice), and initializes a primary population, P_p, with this parameter pair, denoted as (CR_p, F_p). Then, P_p is evolved for a number of iterations,

$$t_{pri} = \alpha \times n,$$

where $\alpha > 1$ is an integer and n stands for the problem's dimension. In [17] the typical case $F, CR \in [0, 1]$ was considered, and the selected parameter values were:

$$\lambda_{CR} = \lambda_F = 0.1, \quad \alpha = 10.$$

After the t_{pri} iterations, three phases are iteratively applied.

The first one is the cloning phase. The primary parameter pair has eight immediate neighboring parameter pairs in the grid, which are defined as:

$$CR' = CR_p + i\,\lambda_{CR}, \quad F' = F_p + j\,\lambda_F, \quad i, j \in \{-1, 0, 1\}, \tag{22.1}$$

where, $i = j = 0$ corresponds to the current primary parameter vector. Each one of the neighboring pairs as well as the primary one are individually assigned to nine secondary populations, P_{s_j}, $j \in \{1, 2, \ldots, 9\}$, which are initialized as clones of the primary population P_p. In order to adapt also the mutation operator, as proposed in [16], four additional secondary populations, also called bridging populations, are used. Each bridging population is a copy of the primary population with same scalar parameters but different mutation operator, selected from the five operators defined in Sect. 22.2.1. The bridging populations are denoted as P_{s_j}, $j \in \{10, \ldots, 13\}$.

The second phase is the performance estimation, where each one of the 13 secondary populations are evolved for a small number of iterations, $t_{sec} \ll t_{pri}$. These short runs aim at probing the local performance dynamic of the secondary populations, providing evidence about the current primary population under the different assigned parameter settings. Typically, t_{sec} shall be significantly smaller than t_{pri} in order to spare computational resources (function evaluations).

For the performance assessment of the secondary populations, the average objective value (AOV) measure was proposed in [17]. The AOV of the secondary population P_{s_j} is defined as:

$$\mathrm{AOV}_j = \frac{1}{N} \sum_{i=1}^{N} f\left(\mathbf{x}_i\right), \tag{22.2}$$

with $\mathbf{x}_i \in P_{s_j}, i = 1, 2, \ldots, N, j = 1, 2, \ldots, 13$. This is the average improvement of the corresponding secondary population with its assigned parameter pair. In order to take into consideration also the crucial issue of diversity, an additional performance measure was considered, namely the objective value standard deviation (OVSD) [16], defined as:

$$\mathrm{OVSD}_j = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f(\mathbf{x}_i) - \mathrm{AOV}_j\right)^2}, \tag{22.3}$$

with $\mathbf{x}_i \in P_{s_j}$, $i = 1, 2, \ldots, N$, $j = 1, 2, \ldots, 13$. Higher values of OVSD are associated to higher diversity, which is preferable for alleviating premature convergence and search stagnation.

Using these two performance metrics, the secondary populations are compared in a Pareto manner. The best among them, i.e., the one with the lowest AOV and the highest OVSD, is selected as the new primary population along with its parameters. Note that more than one non-dominated populations may appear. Since these populations are incomparable, one is randomly selected among them to become the new primary population. Moreover, the concept of sufficient improvement can be implemented by replacing the primary population with a new one only if AOV is improved for at least some $\varepsilon > 0$, otherwise the current primary population is retained.

Eventually, the dynamic's deployment phase takes place, where the selected new primary population is evolved for t_{pri} iterations to reveal its dynamic with the adopted parameters. This step completes a full cycle of GPAM applied on DE. Following the notation in [16], the combination of DE with GPAM is denoted as DEGPOA. For further details on the method the reader is referred to [16, 17].

22.3 Experimental Analysis

In [20] a preliminary study on the impact of the three parameters of GPAM, namely t_{pri}, t_{sec}, $\lambda = \lambda_F = \lambda_{CR}$, on DEGPOA was presented. The study was based on simplistic experimental configuration where one of the parameters was changed at a time, keeping the rest fixed. The considered parameter levels were as reported in Table 22.1. In the present study, we extend the previous results with a full factorial design investigation. The produced $4^3 = 64$ DEGPOA instances are named after the values of the three parameters (n is skipped for t_{pri}) as follows:

$$t_{sec}\_t_{pri}\_\lambda.$$

Table 22.1 The considered parameter values in our experimental setting

Parameter	Level			
	1	2	3	4
t_{sec}	5	10	15	20
t_{pri}	$5n$	$10n$	$15n$	$20n$
λ	0.05	0.1	0.15	0.2

For example, `10_5_0.15` stands for the instance with $t_{sec} = 10$, $t_{pri} = 5n$, and $\lambda = 0.15$.

All experiments were conducted on the state-of-the-art CEC 2013 test suite [1], strictly following its guidelines. The test suite consists of 28 unimodal, multimodal, and composite functions, henceforth denoted as f_1-f_{28}. The most common dimensions $n = 10$ and $n = 30$ were considered, while the search space for all test problems was $[-100, 100]^n$. As dictated by the test suite, the maximum computational budget was set to $T_{\max} = 10^4 \times n$ function evaluations. Solution quality was measured according to the error gap of the detected solution from the known optimal one. Fixed population size was used in all experiments, while 51 independent runs per test problem were conducted.

The preliminary statistical analysis in [20] offered clear indications that t_{sec} is the most influential parameter, followed by λ and t_{pri}. The reader is referred to [20] for a detailed explanation. The present work aims at expanding the previous analysis through a full factorial design that can reveal possible parameter interactions and correlations. For this reason, our current experimental configuration reported above is identical to the previous one in [20].

The first part of our analysis was based on 3-way ANOVA on all the 64 DEGPOA instances, based on their average error values over the whole test suite.[1] The resulting ANOVA tables for both dimensions are reported in Table 22.2. The tables contain the typical ANOVA invormation, with the last column (marked as "Prob > F") providing the p-values. For the 10-dimensional case (upper part of Table 22.2), we can see p-values smaller than 0.05 for t_{sec} and λ, which suggest significant impact of these two parameters, in contrast to t_{pri}. However, there seems to be no statistically significant interactions between any two of the parameters. This is in line with the results in the preliminary analysis in [20].

In the 30-dimensional case, the impact of t_{sec} remains significant, but now t_{pri} exhibits significant impact instead of λ. This interesting evidence suggests that, as dimension increases, the granularity of the grid becomes less important than the effort spent for the deployment of the algorithm's dynamic. Also, it shows that the computational budget devoted to the estimation of the best parameter setting through the secondary populations is always important. This is a reasonable effect because t_{sec} is the key-factor for proper assessment of the secondary populations and, consequently, for effective grid search in the parameter domain.

In addition to the these evidence, we followed an alternative analysis to identify the most promising parameter combinations. For this purpose, we employed Wilcoxon rank-sum tests to compare all algorithm instances among them on each test function, individually, based on their solution errors in the 51 independent runs. For each comparison between two algorithm instances, the rank-sum test revealed possible statistically significant difference between the compared error samples. In this case, the algorithm with the smallest mean error was awarded a win, and the opponent algorithm a loss. In case of insignificant difference, both algorithms assumed a draw.

[1] The MathWorks Matlab® software was used for this purpose.

Table 22.2 Three-way ANOVA for the 64 DEGPOA instances

Source	Sum sq.	d.f.	Mean sq.	F	Prob > F
Dimension: $n = 10$					
t_{sec}	12.38	3	4.12667	6.15	0.0025
t_{pri}	3.1402	3	1.04675	1.56	0.2219
λ	13.7274	3	4.5758	6.82	0.0014
$t_{sec} * t_{pri}$	8.2554	9	0.91727	1.37	0.251
$t_{sec} * \lambda$	4.5431	9	0.50478	0.75	0.6592
$t_{pri} * \lambda$	2.8411	9	0.31568	0.47	0.8815
$Error$	18.1128	27	0.67084		
$Total$	63	63			
Dimension: $n = 30$					
t_{sec}	15.949	3	5.31632	8.6	0.0004
t_{pri}	9.1731	3	3.05771	4.95	0.0073
λ	3.7093	3	1.23643	2	0.1376
$t_{sec} * t_{pri}$	4.9029	9	0.54477	0.88	0.5534
$t_{sec} * \lambda$	6.8171	9	0.75746	1.23	0.3208
$t_{pri} * \lambda$	5.7618	9	0.6402	1.04	0.4382
$Error$	16.6868	27	0.61803		
$Total$	63	63			

After all pairwise comparisons, we calculated a rank measure for each algorithm, defined as the difference between its wins and losses:

$$\texttt{rank}(\text{alg}) = \text{wins}_{\text{alg}} - \text{losses}_{\text{alg}}.$$

Since each algorithm is compared to 63 others on 28 functions, it holds that:

$$-1764 \leqslant \texttt{rank}(\text{alg}) \leqslant 1764,$$

with higher values denoting better performance (higher number of wins against losses).

All ranks are sorted and graphically illustrated in Fig. 22.1. The sorted ranks clearly illustrate that the performance among all algorithm instances varies significantly in some cases. In order to get a better picture, we isolated the algorithm instances with positive ranks and conducted the same rank analysis strictly among them. The corresponding sorted ranks are given in Fig. 22.2.

As we can see, the top-performing instances for both dimensions prefer larger values of t_{pri} ($15n$ or $20n$) but smaller values of t_{sec} (typically 5 or 10). Interpreting this evidence, we conclude that the most efficient algorithms are the ones putting more emphasis on the dynamic deployment phase rather than the performance estimation

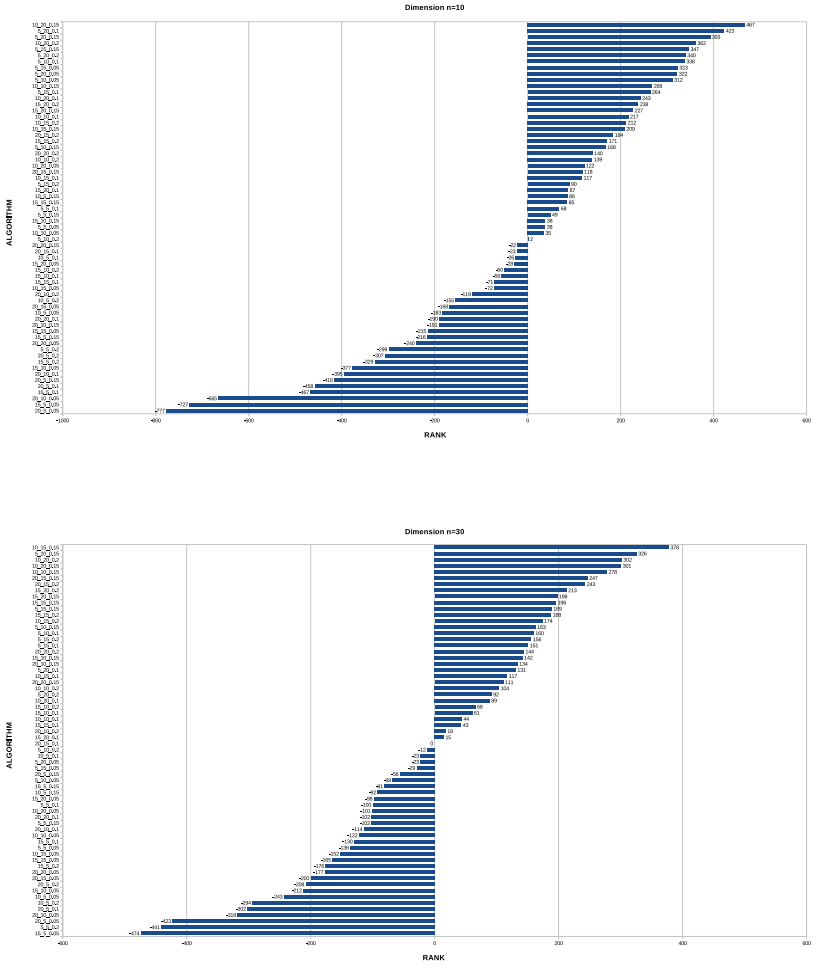

Fig. 22.1 Sorted ranks of the 64 DEPGPOA instances

phase, where rough estimations are adequate to guide the grid search. Interestingly, higher levels of λ appear more frequently in the top-performing algorithms. This shows that the DEGPOA approach does not require highly fine-grained grids, probably as a consequence of DE's reduced sensitivity in marginal changes of its scalar parameters (this finding has been reported also in [16]).

The observed patterns of higher t_{pri} and λ levels and lower t_{sec} levels are more clearly illustrated in Figs. 22.3 and 22.4. These figures illustrate all algorithm instances sorted according to their ranks, along with their parameter levels. In order to avoid scaling issues, the level indices (i.e., 1, 2, 3, 4) of the parameter values are used instead of their actual values. The aforementioned pattern is clearly observed on the right part of the figures, where the best-performing algorithm instances lie, verifying the previous findings.

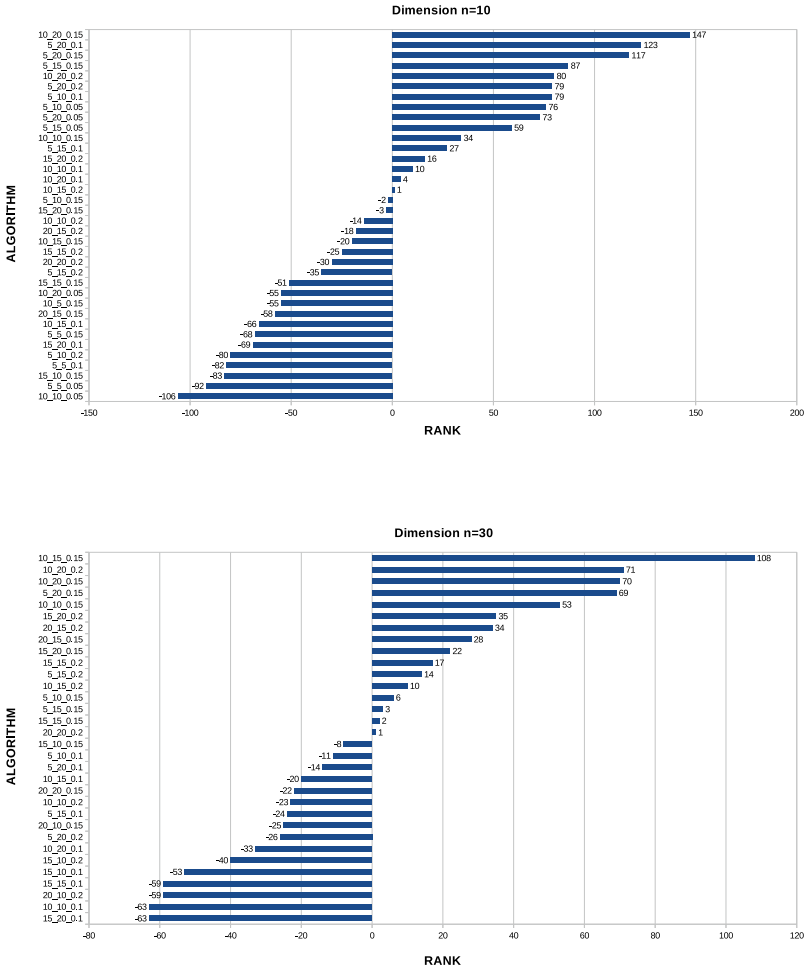

Fig. 22.2 Sorted ranks of the 64 DEPGPOA instances

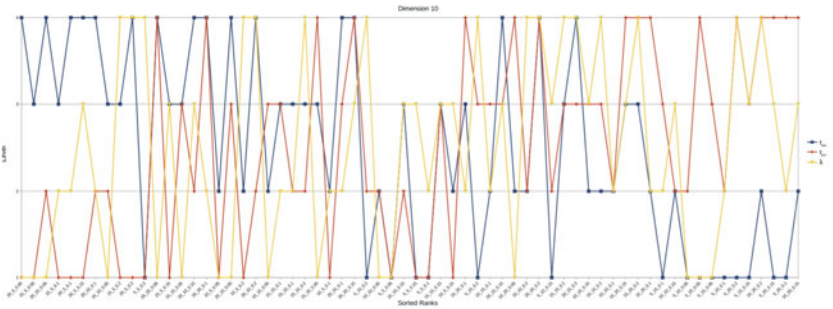

Fig. 22.3 The 64 DEGPOA instances sorted by their ranks, along with their corresponding parameter levels for the 10-dimensional functions

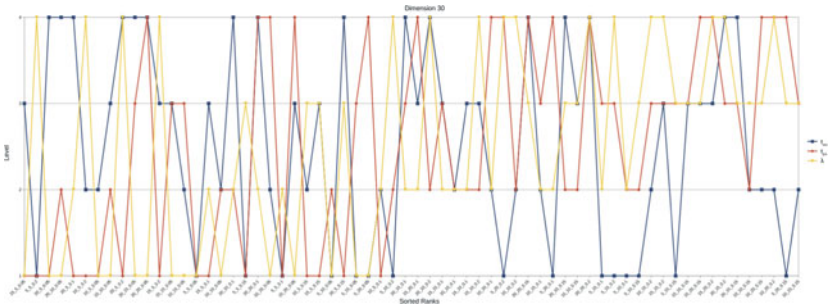

Fig. 22.4 The 64 DEGPOA instances sorted by their ranks, along with their corresponding parameter levels for the 30-dimensional functions

22.4 Conclusion

We presented an extended analysis of GPAM applied on the DE algorithm, adding to previous experimental analysis. The presented study is based on full factorial statistical analysis for various parameter levels of the resulting DEGPOA approach, based on the established CEC 2013 test suite.

The main findings can be summarized as follows:

(a) In smaller dimension, t_{sec} and λ have higher impact on the algorithms than t_{pri}. In higher dimension, t_{sec} and t_{pri} exhibit higher statistical significance than λ. Thus, as dimension increases, the granularity of the grid becomes less important than the deployment of the algorithm's dynamic.
(b) Higher values of t_{pri} and λ, and lower values of t_{sec} are associated with the best performing algorithm instances.

These findings verify at large previous findings and they justify the parameter value choices in previous works on DEGPOA.

Naturally, the observed results are highly associated with the specific algorithm and testbed. Further experimentation is needed to probe the algorithm's behavior in different experimental environments, as well as for different algorithms.

References

1. Complementary material: Special session & competition on real-parameter single objective optimization at CEC'2013, http://www.ntu.edu.sg
2. T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation* (Springer, Berlin, 2006)
3. M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective* (Springer, Berlin, 2009)
4. J. Brest, M.S. Maucec, Self-adaptive differential evolution algorithm using population size reduction and three strategies. Soft Comput. **15**, 2157–2174 (2011)
5. S. Das, P.N. Suganthan, Differential evolution: A survey of the state-of-the-art. IEEE Trans. Evol. Comput. **15**(1), 4–31 (2011)

6. A.E. Eiben, R. Hinterding, Z. Michalewicz, Parameter control in evolutionary algorithms. IEEE Trans. Evol. Comput. **3**(2), 124–141 (1999)
7. A.E. Eiben, S.K. Smit, Evolutionary algorithm parameters and methods to tune them, in *Autonomous Search*, chapter 2, eds. by Y. Hamadi, E. Monfroy, F. Saubion (Springer, Berlin, 2011), pp. 15–36
8. M. Gendreau, J. Potvin, *Handbook of Metaheuristics*, 2nd edn. (Springer, New York, 2010)
9. A. Gogna, A. Tayal, Metaheuristics: review and application. J. Exp. Theor. Artif. Intell. **25**(4), 503–526 (2013)
10. H.H. Hoos, Automated algorithm configuration and parameter tuning, in *Autonomous Search*, chapter 3, eds. by Y. Hamadi, E. Monfroy, F. Saubion (Springer, Berlin, 2011), pp. 37–72
11. F. Hutter, H.H. Hoos, K. Leyton-Brown, Sequential model-based optimization for general algorithm configuration, in *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy. Selected Papers*, ed. by A.C. Coello Coello (Springer, Berlin, 2011), pp. 507–523
12. K.V. Price, R.M. Storn, J.A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization* (Springer, Berlin, 2005)
13. A.K. Qin, V.L. Huang, P.N. Suganthan, Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans. Evol. Comput. **13**(2), 398–417 (2009)
14. R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Global Opt. **11**, 341–359 (1997)
15. R. Tanabe, A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in *2014 IEEE Congress on Evolutionary Computation* (2014)
16. V.A. Tatsis, K.E. Parsopoulos, Grid search for operator and parameter control in differential evolution, in *9th Hellenic Conference on Artificial Intelligence*, SETN '16 (ACM, 2016), pp. 1–9
17. V.A. Tatsis, K.E. Parsopoulos, Differential evolution with grid-based parameter adaptation. Soft Comput. **21**(8), 2105–2127 (2017)
18. V.A. Tatsis, K.E. Parsopoulos. Grid-based parameter adaptation in particle swarm optimization, in *12th Metaheuristics International Conference (MIC 2017)* (2017), pp. 217–226
19. V.A. Tatsis, K.E. Parsopoulos, Experimental assessment of differential evolution with grid-based parameter adaptation. Int. J. Artif. Intell. Tools **27**(04), 1–20 (2018)
20. V.A. Tatsis, K.E. Parsopoulos, On the sensitivity of the grid-based parameter adaptation method, in *7th International Conference on Metaheuristics and Nature Inspired Computing (META 2018)* (2018), pp. 86–94
21. J. Torres-Jiménez, J. Pavón, Applications of metaheuristics in real-life problems. Prog. Artif. Intell. **2**(4), 175–176 (2014)
22. J. Zhang, A.C. Sanderson, JADE: Adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**, 945–958 (2009)