International Journal on Artificial Intelligence Tools
Vol. 27, No. 4 (2018) 1860004 (20 pages)
© World Scientific Publishing Company
DOI: 10.1142/S0218213018600047

Experimental Assessment of Differential Evolution with Grid-Based Parameter Adaptation

Vasileios A. Tatsis and Konstantinos E. Parsopoulos
Department of Computer Science & Engineering
University of Ioannina, GR-45110 Ioannina, Greece,
{*vtatsis,kostasp*}*@cse.uoi.gr*

Received 31 August 2017
Accepted 9 February 2018
Published 27 June 2018

Evolutionary algorithms have been long established as an essential field of research in Computational Intelligence. Differential Evolution is placed among the most successful algorithms of this type. However, it has proved to be highly sensitive on its parameters. For this purpose, offline and online parameter-control methods have been proposed. Recently, a grid-based parameter adaptation procedure was introduced and successfully applied on Differential Evolution. Despite the generality of the method, the resulting algorithm was capable to compete with already tuned adaptive algorithms on high-dimensional test problems without any additional preprocessing. The present work extends the experimental study of this approach on the state-of-the-art CEC-2013 test suite. Two variants of the algorithm are considered and different initial conditions are tested to shed light on its performance aspects. Comparisons with other algorithms as well as between the proposed approaches are reported. The results verify the potential of the grid-based parameter adaptation method as a general-purpose alternative for parameter setting.

Keywords: Metaheuristic optimization; differential evolution; parameter control; grid search.

1. Introduction

Modern technological developments have created the necessity for autonomous systems that involve optimization problems of high complexity. The solution of such problems requires optimization algorithms capable of providing acceptable solutions in reasonable time with minimal human intervention or modeling constraints. Metaheuristics[17] constitute a wide category of algorithms that satisfy these necessities.[18,35] Evolutionary algorithms[14] stand in a salient position among metaheuristics. They have been intensely investigated and evolved in the past 50 years to fit the requirements of contemporary problems. Their simplicity and effectiveness has resulted in popular state-of-the-art paradigms, including Genetic Algorithms,[24] Evolution Strategies,[4] Particle Swarm Optimization,[26] and Differential Evolution.[27]

Evolutionary algorithms are based on stochastic search procedures controlled by a number of parameters. Proper parameter values allow the adaptation of the algorithm dynamics to the problem at hand, promoting the desirable exploration/exploitation behavior. However, they also impose a number of decisions to be made by the user. In this context, some algorithms exhibit significant parameter sensitivity, Differential Evolution being a typical example. In such cases, parameter-setting methods become extremely important. A significant number of diverse methods have been proposed for this purpose. They can be distinguished in two basic categories, namely the *offline* parameter tuning prior to the algorithm's execution, and the *online* parameter adaptation (control) based on the algorithm's feedback during its execution.[13]

Offline parameter tuning is typically based on a preprocessing phase of intense experimentation, where alternative parameter settings are evaluated on prespecified sets of test problems. It can be viewed as a training phase, aiming at revealing promising levels of the considered parameters for the specific problem types. Such procedures can be effective when they are provided a representative set of test problems, diverse parameter settings, and adequate computational resources. Popular approaches of this type are the Design of Experiments,[3] F-Race,[5] and ParamILS.[19] An obvious drawback of these approaches is their computational requirements that may be comparable or even exceed the ones required for solving the actual optimization problem. Another drawback is the possible over-specialization of the detected parameter values, which probably renders the algorithm ineffective in problems of different type.

On the other hand, online parameter control utilizes feedback from the algorithm and adapts the parameters during its execution. Thus, the parameters are adapted in real-time without requiring the laborious preprocessing phase. Naturally, the derived parameter setting may not be reusable even in similar problems. Nevertheless, this procedure requires only minimal user intervention, thereby avoiding user-induced biases. Typical representatives of online tuners are the dynamic parameter adaptation approaches presented in Refs. 13 and 19.

Recently, an online grid-based parameter adaptation method was proposed.[32] It is based on a local search procedure within a grid determined by the discretization of the parameters' search space. The method was initially validated on Differential Evolution for the online control of its scalar parameters and its crossover type. In subsequent study, it was extended to control also the mutation operator.[33] Note that this problem itself defines a mixed-integer optimization task. The grid-based approach has enriched the existent Differential Evolution literature on parameter tuning and adaptation.[8,28,30,31,36–41] In addition, it avoids over-specialization on the specific algorithm, providing a rather general framework applicable also to other metaheuristics.[34]

In previous works,[32,33] the grid-based parameter adaptation method was validated on a high-dimensional test suite proposed in the special issue on Large Scale

Continuous Optimization Problems of the Soft Computing journal.[23] This choice was reasonable for initial validation because dimensionality has proved to exponentially increase the difficulty of optimization problems.[10] Naturally, parameter adaptation becomes indispensable particularly in such demanding cases. Nevertheless, a variety of specialized test suites of moderate and low dimension has been recently developed for benchmarking of evolutionary algorithms. Perhaps the most popular ones are the CEC test suites, proposed in the series of the IEEE International Congress on Evolutionary Computation conferences. The CEC-2013 test suite[21] is currently considered as mainstream standard for validating and comparing evolutionary algorithms.

The present work builds on previous studies[32,33] to validate Differential Evolution with grid-based parameter adaptation on the CEC-2013 test suite. Specifically, the mutation operator and the two scalar parameters of the algorithm are adapted based on the grid search scheme. Two variants are studied, using one and two performance measures of the algorithm, respectively. The resulting algorithms are characterized by the following desirable properties:

(a) They are completely autonomous from the user during their execution.
(b) They are exempted from the necessity for long preprocessing times, thereby sparing significant amounts of computational time and resources.
(c) They can ameliorate the performance of the algorithm when unsuitable initial parameter values are selected by the user.

Comparisons between the proposed approaches and various other algorithms on the CEC-2013 test suite show that Differential Evolution equipped with the grid-based parameter adaptation method can become highly competitive even against algorithms that have been *a priori* tuned on the specific test suite. Statistical analysis of the results is conducted in all cases.

The rest of the paper is organized as follows: Section 2 offers a brief outline of the Differential Evolution algorithm. The studied grid-based parameter adaptation method is presented in Section 3. Experimental analysis on the CEC-2013 test suite is reported in Section 4. The paper concludes in Section 5.

2. Differential Evolution

Differential Evolution (DE) was originally introduced by Storn and Price.[29] It is a population-based search algorithm that integrates three evolutionary procedures, namely *mutation*, *recombination*, and *selection*. Mutation produces new candidate solutions through linear combinations of two or more randomly selected members of the population. Stochastic recombination produces new trial points by combining components of the mutated and the original ones. These points compete with the current members of the population in order to produce a new population. Differential Evolution has recently gained increasing popularity, counting a significant number of variants and applications.[11]

Table 1. Common mutation operators of Differential Evolution.

Notation	Algorithm	Mutation Operator
DE_1	DE/best/1	$u_i^{(t+1)} = x_g^{(t)} + \alpha\left(x_{r_1}^{(t)} - x_{r_2}^{(t)}\right)$
DE_2	DE/rand/1	$u_i^{(t+1)} = x_{r_1}^{(t)} + \alpha\left(x_{r_2}^{(t)} - x_{r_3}^{(t)}\right)$
DE_3	DE/current-to-best	$u_i^{(t+1)} = x_i^{(t)} + \alpha\left(x_g^{(t)} - x_i^{(t)} + x_{r_1}^{(t)} - x_{r_2}^{(t)}\right)$
DE_4	DE/best/2	$u_i^{(t+1)} = x_g^{(t)} + \alpha\left(x_{r_1}^{(t)} - x_{r_2}^{(t)} + x_{r_3}^{(t)} - x_{r_4}^{(t)}\right)$
DE_5	DE/rand/2	$u_i^{(t+1)} = x_{r_1}^{(t)} + \alpha\left(x_{r_2}^{(t)} - x_{r_3}^{(t)} + x_{r_4}^{(t)} - x_{r_5}^{(t)}\right)$

Putting it formally, let the general form of the n-dimensional unconstrained minimization problem

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

where X is the search space defined as an n-dimensional hyperbox. Differential Evolution employs a population of N search points (agents),

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

Each member of the population is a candidate solution of the problem,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \in X\,, \quad i \in I\,,$$

where the set I denotes the set of indices,

$$I = \{1, 2, \ldots, N\}\,.$$

The population is randomly and uniformly initialized in X. Then, it is iteratively evolved according to three basic procedures.

The first procedure is *mutation*, where a new (mutated) vector is produced for each member of the population $P^{(t)}$, with t denoting the iteration number. Let $x_i^{(t)}$ denote the ith individual at iteration t, and $u_i^{(t+1)}$ denote the produced mutated vector. There is a variety of alternative mutation operators. Among the most popular ones are the mutation operators reported in Table 1. The parameter $\alpha \in (0, 1]$, also called the *scale factor*, controls the step size of the mutation.[11] The index g denotes the best individual in the current population, i.e., the one with the smallest objective value in $P^{(t)}$, while the indices r_j, $j = 1, 2, \ldots, 5$, are mutually different and randomly selected from the set $I_i = I \backslash \{i\}$.

Mutation is followed by *crossover*, where a new *trial vector* $v_i^{(t+1)}$ is produced for each individual by inheriting components from the original vector and the mutated one, as follows,

$$v_{ij}^{(t+1)} = \begin{cases} u_{ij}^{(t+1)}, & \text{if } (\texttt{rand}() \leqslant \beta) \;\texttt{OR}\; (j = RI(D,i)),\,, \\ x_{ij}^{(t)}\,, & \text{otherwise}\,, \end{cases} \tag{1}$$

where $j \in D = \{1, 2, \ldots, n\}$; $\texttt{rand}(\cdot)$ is a uniform random number generator in the range $[0, 1]$; $\beta \in (0, 1]$ is another parameter of the algorithm, also called the *crossover rate*; and $RI(D, i)$ is an integer selected randomly from D for the ith individual. The crossover rate controls the probability of favoring components of the mutated vector than the original one, while the second condition of Eq. (1) ensures that the trial vector inherits at least one component from the mutated vector. This crossover operator is also called *binomial crossover*.

An alternative is the *exponential crossover* where, starting from a random position $k \in D$, the components of $u_i^{(t+1)}$ are copied into $v_i^{(t+1)}$ until a stochastic condition is satisfied. When the condition fails, the procedure stops and the rest of the components are inherited from the original vector $x_i^{(t)}$.[27]

Eventually, *selection* takes place where $v_i^{(t+1)}$ competes against $x_i^{(t)}$ for its inclusion in the new population, i.e.,

$$x_i^{(t+1)} = \begin{cases} v_i^{(t+1)}, & \text{if } f\left(v_i^{(t+1)}\right) \leqslant f\left(x_i^{(t)}\right), \\ x_i^{(t)}, & \text{otherwise}. \end{cases} \tag{2}$$

The algorithm iterates until a stopping criterion is satisfied. This criterion is usually related to the maximum acceptable number of iterations or the desirable solution quality. The best detected solution is reported at termination.

3. Grid-Based Parameter Adaptation Method

The *Differential Evolution with Grid-Based Parameter Adaptation* (DEGPA) algorithm[32] was initially proposed for controlling the scalar parameters α and β of the standard Differential Evolution. It was further extended to the DEGPOA[33] approach that dynamically adapts also the mutation operator among the ones reported in Table 1. The main idea in DEGPA is the discretization of the parameters domain. Let S_α and S_β denote the desirable ranges of the parameters α and β, respectively. Then, a grid G is formed by discretizing the set $S_\alpha \times S_\beta$, by using fixed discretization steps λ_α and λ_β, respectively.[32]

The parameters of the algorithm are initialized on the central point of the grid and the population is randomly initialized. The population is also called the *primary population*. Then, the algorithm iterates the following phases[32]:

(a) *Dynamics deployment phase*: the primary population is evolved for t_p iterations using its parameters (α_p, β_p) and then stops.

(b) *Cloning phase*: the evolved primary population is reproduced to nine secondary populations, each one assuming an immediate neighboring parameter pair in the grid, i.e.,

$$(\alpha_i, \beta_j) = (\alpha_p + i\,\lambda_\alpha, \beta_p + j\,\lambda_\beta), \quad i, j \in \{-1, 0, 1\}. \tag{3}$$

(c) *Performance estimation phase*: Each secondary population is evolved for a small number of iterations, $t_s \ll t_p$, in order to reveal the population's dynamics under the corresponding parameter pair.

Algorithm 1 Pseudocode of the DEGPOA approach.

```
1: $(P_p, o_p, \alpha_p, \beta_p) \leftarrow$ **initialization**( )   // primary population, operator, and parameters
2: $m \leftarrow 13$   // number of secondary populations
3: **while** (not termination) **do**
4:     /* DYNAMICS DEPLOYMENT PHASE */
5:     **evolve**$(P_p, o_p, \alpha_p, \beta_p, t_p)$
6:     /* CLONING */
7:     **for** $(s = 1 \ldots m)$ **do**
8:         $P_s \leftarrow P_p$   // copy primary to secondary population
9:         **if** $(s \leqslant 9)$ **then**
10:            /* Normal secondary population */
11:            $o_s \leftarrow o_p$   // retain mutation operator
12:            **assign**$(P_s, o_s, \alpha_s, \beta_s)$   // use Eq. (3) for new parameter pair
13:        **else**
14:            /* Bridging secondary population */
15:            $o_s \leftarrow$ **new_operator**( )   // change mutation operator
16:            $(\alpha_s, \beta_s) \leftarrow (\alpha_p, \beta_p)$   // retain parameters pair
17:        **end if**
18:        /* PERFORMANCE ESTIMATION */
19:        **evolve**$(P_s, o_s, \alpha_s, \beta_s, t_s)$
20:    **end for**
21:    /* UPDATE PRIMARY POPULATION */
22:    $P_{\text{best}} \leftarrow$ **best_population**$(P_1, \ldots, P_{13})$
23:    **if** $\left(\bar{f}_p - \bar{f}_{\text{best}} \geqslant \varepsilon_{\min}\right)$ **then**   // adequate improvement achieved
24:        $(P_p, o_p, \alpha_p, \beta_p) \leftarrow (P_{\text{best}}, o_{\text{best}}, \alpha_{\text{best}}, \beta_{\text{best}})$
25:    **end if**
26: **end while**
```

(d) *Update primary population*: The performance of each secondary population is assessed. The best one is selected as the new primary population along with its parameters.

The basic DEGPA scheme was extended in DEGPOA[33] by considering also adaptive mutation operator. The search space for the mutation operator is discrete and lacks of specific ordering. For this reason, the DEGPOA scheme assumes five additional secondary populations (called *bridging populations*), each one adopting the primary parameter pair (α_p, β_p) but a different mutation operator from Table 1. The rest of DEGPOA follows closely the procedures of DEGPA.[33]

An issue of interest is the assessment criterion for the secondary populations after the performance estimation phase. In DEGPA, the average objective values $\bar{f}_s$, $s = 1, 2, \ldots, 9$, of the secondary populations are computed. The best secondary population replaces the primary one if it achieves adequate improvement.[32] The same criterion was initially adopted in DEGPOA,[33] and it was further enhanced in the eDEGPOA variant[33] by considering an additional performance measure for each secondary population, namely the standard deviation σ_s of the objective values. This modification aims at promoting diversity-preserving populations in order to avoid search stagnation. In this case, the selection of the best secondary population is based on Pareto dominance of the pairs $(\bar{f}_s, \sigma_s)$.[33]

Pseudocode for DEGPOA is provided in Algorithm 1. In the pseudocode, o_p and o_s denote mutation operators from Table 1 (the ordering is irrelevant), and $\varepsilon_{\min}$ is a minimal improvement threshold for updating the parameters of the primary population. The reader is referred to the original works[32,33] for a more thorough presentation.

4. Experimental Evaluation

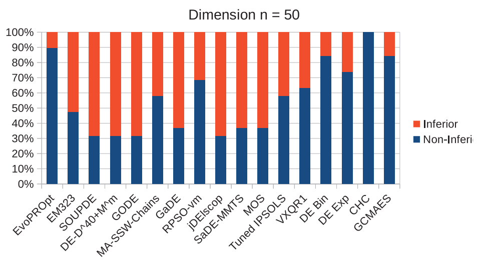
The DEGPOA and eDEGPOA approaches have been validated on a high-dimensional test suite,[32,33] proposed in the Special Issue on Large Scale Continuous Optimization Problems of the Soft Computing journal.[23] Our primary interest in high-dimensional problems stemmed from the fact that, in most cases, their complexity is exponentially increased with dimension. Consequently, the tuning of a metaheuristic becomes extremely laborious and resource consuming.

Figure 1 summarizes comparisons of the studied approaches with other algorithms, as they are reported in Ref. 33. Specifically, DEGPOA and eDEGPOA were compared with each other on the basis of their solutions quality in 25 experiments. The figures report the percentage of problems where DEGPOA and eDEGPOA were non-inferior (better or statistically equivalent) or inferior to the other algorithms. Note that the results of the other algorithms were adopted from the relevant repository of the test suite,[1] and refer to versions already tuned for that test suite, contrary to DEGPOA and eDEGPOA that did not undergo any tuning preprocessing.

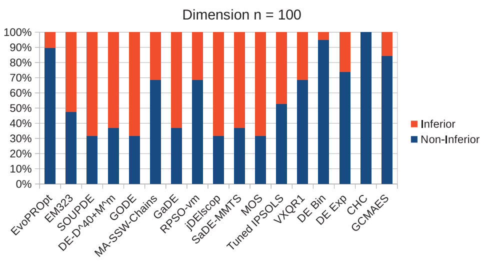
Although high-dimensional problems appear to be the natural application field of the studied approaches, their assessment would be incomplete without considering any of the mainstream test suites for validation of evolutionary algorithms despite that their test problems have lower dimension. The present work aims at filling this gap. For this purpose, the mainstream CEC-2013[21] test suite was selected. It consists of 28 benchmark problems, henceforth denoted as f_1-f_{28}, which include unimodal, multimodal, and composite functions. The corresponding search spaces are fixed to $[-100, 100]^n$ for all test problems, where n stands for the problem dimension. Although lower dimensions are also proposed in the test suite, we considered only the challenging $n = 30$ and $n = 50$ cases.

The available computational budget, as dictated by the test suite, was equal to

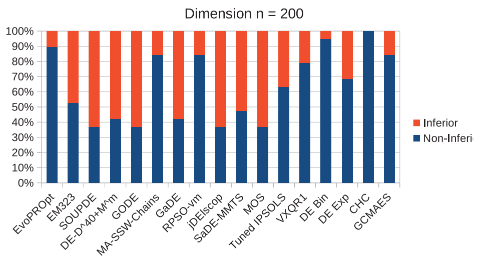
$$T_{\max} = n \times 10^4 \,,$$



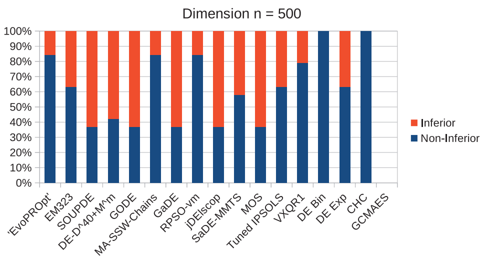
(a)



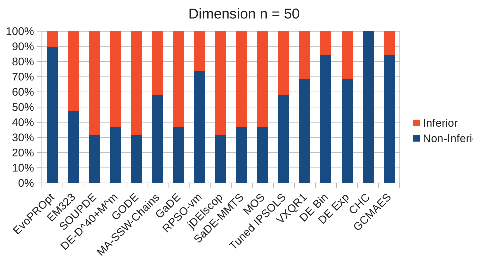
(b)



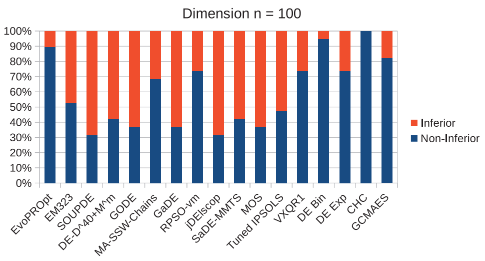
(c)



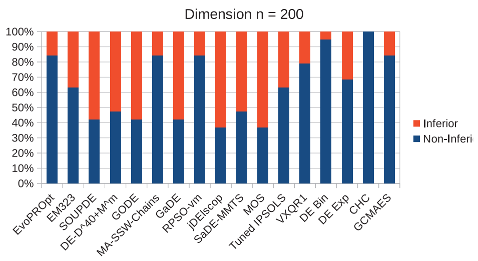
(d)



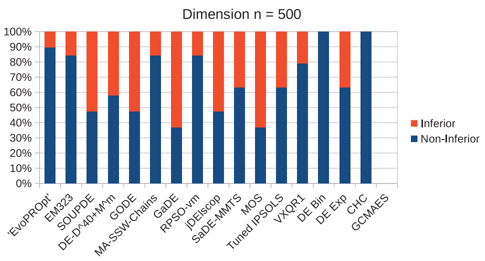
(e)



(f)



(g)



(h)

Fig. 1. Percentage of test problems where DEGPOA (Figs. (a)–(d)) and eDEGPOA (Figs. (e)–(h)) were non-inferior (statistically better or equivalent) or inferior to the competing algorithms.

function evaluations. The performance criterion for the algorithms was the error gap between the (known) optimal solution x_{opt} and the solution x^* achieved by the algorithm, i.e.,

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

Following the test suite's setting,[21] fixed population size $N = 60$ was used for the algorithms, and 51 independent experiments were conducted per problem and algorithm. We closely followed the settings of the test suite in order to achieve results comparable to the rest of the algorithms reported in Refs. 21 and 22.

The initial primary parameter pair of DEGPOA and eDEGPOA was set at the central grid point

$$(\alpha, \beta) = (0.5, 0.5)\,,$$

while the initial mutation operator at each experiment was randomly selected from the ones in Table 1. These variants are henceforth denoted as $\text{DEGPOA}_{0.5}$ and $\text{eDEGPOA}_{0.5}$, respectively. Additionally, the distant initial pairs

$$(a, b) = (0.2, 0.2)\,, \qquad (a, b) = (0.8, 0.8)\,,$$

were also considered to investigate possible performance fluctuations related to the initial settings. We henceforth denote the corresponding algorithms as $\text{DEGPOA}_{0.2}$, $\text{eDEGPOA}_{0.2}$, $\text{DEGPOA}_{0.8}$, and $\text{eDEGPOA}_{0.8}$, respectively. In all cases, exponential crossover was used according to our previous analysis.[32] Also, the values $t_s = 5$, and $\varepsilon_{\min} = 10^{-2}$ (optional parameter) were adopted from Ref. 32, along with the linearly increasing t_p in the range $[10\,n, 14\,n]$.

All DEGPOA and eDEGPOA variants were applied on the CEC-2013 test suite according to the aforementioned settings, and their results were recorded and statistically analyzed in order to facilitate comparisons with a number of adaptive and non-adaptive algorithms. The results for the rest of the algorithms were adopted directly from the relevant sources.[2] However, it is important to note that these results refer to tuned versions of the algorithms without taking into account the computational cost required for their tuning. On the other hand, DEGPOA and eDEGPOA do not need any preprocessing or preliminary experimentation. Moreover, the computational budget for the CEC-2013 problems is quite restrictive. This renders the comparisons even more challenging.

The performance comparisons with competitive algorithms reported in the relevant repository of the CEC-2013 test suite[2] were based on statistical Wilcoxon tests at confidence level 95% of the achieved solution errors between DEGPOA, eDEGPOA, and the following 10 algorithms: SMADE,[9] TLBSaDE,[6] JANDE,[7] DE_APC,[16] TPC-GA,[15] PVADE,[12] CDASA,[20] and PLES.[25] For completeness purpose, all standard variants of Differential Evolution reported in Table 1 were also included in the competing algorithms. At each comparison, a win was counted for DEGPOA or eDEGPOA whenever it achieved statistically superior performance than the competitor. In the opposite case a loss was counted, while statistically insignificant differences between algorithms were considered as ties.

Tables 2 and 3 report the number of wins (denoted as "+"), losses (denoted as "−"), and ties (denoted as "=") for DEGPOA and eDEGPOA, respectively, against the corresponding standard DE algorithms. Note that each standard DE algorithm adopted the same parameter values as the initial parameter setting of

Table 2. Statistical comparisons between DEGPOA and standard DE on the CEC-2013 test problems.

	Dimension					
	30			50		
DEGPOA$_{0.2}$ vs	+	−	=	+	−	=
DE$_1$	16	4	8	15	4	9
DE$_2$	12	8	8	14	8	6
DE$_3$	19	3	6	20	6	2
DE$_4$	12	6	10	12	8	8
DE$_5$	12	7	9	14	9	5
DEGPOA$_{0.5}$ vs	+	−	=	+	−	=
DE$_1$	18	1	9	18	2	8
DE$_2$	18	5	5	22	3	3
DE$_3$	17	5	6	17	5	6
DE$_4$	19	4	4	20	3	5
DE$_5$	19	5	4	24	3	1
DEGPOA$_{0.8}$ vs	+	−	=	+	−	=
DE$_1$	13	6	9	14	9	5
DE$_2$	15	6	7	18	6	4
DE$_3$	13	7	8	16	7	5
DE$_4$	20	4	4	19	5	4
DE$_5$	20	3	5	22	2	4

"+" denotes wins; "−" denotes losses; "=" denotes ties.

the competing DEGPOA or eDEGPOA approach. As we can see in Table 2, the DEGPOA approaches outperformed all standard DE algorithms regardless of the initial parameter setting. This indicates that the observed improvements from the use of the grid-based parameter adaptation are not related to the initial parameter setting. Instead, DEGPOA was capable of tuning the algorithm regardless of the initial parameters and operator, achieving far better results than the corresponding DE algorithm with the same parameters and mutation operator.

Interestingly, we can also see that the average number of wins for DEGPOA increases with dimension. Indeed, for the 50-dimensional problems the average numbers of wins for the three DEGPOA algorithms is equal to 15.0, 20.2, and 17.8, while the corresponding numbers for the 30-dimensional problems are 14.2, 18.2, and 16.2. This evidence suggests that the search stagnation of DE in higher dimensions can be ameliorated through the proposed parameter tuning. Similar results were obtained for eDEGPOA, with average numbers of wins equal to 14.2, 18.0, and 19.6 for the 30-dimensional problems, and 20.6, 18.0, and 21.6, for the 50-dimensional cases. All the average numbers of wins are illustrated in Fig. 2.

In the same vein, Tables 4 and 5 report results from statistical comparisons of DEGPOA and eDEGPOA with other algorithms. Among them are included

Table 3. Statistical comparisons between eDEGPOA and standard DE on the CEC-2013 test problems.

	Dimension					
	30			50		
eDEGPOA$_{0.2}$ vs	+	−	=	+	−	=
DE$_1$	14	2	12	21	2	5
DE$_2$	12	7	9	20	4	4
DE$_3$	20	3	5	21	6	1
DE$_4$	13	7	8	20	2	6
DE$_5$	12	9	7	21	4	3
eDEGPOA$_{0.5}$ vs	+	−	=	+	−	=
DE$_1$	16	1	11	15	1	12
DE$_2$	18	4	6	20	3	5
DE$_3$	15	3	10	14	5	9
DE$_4$	21	3	4	18	3	7
DE$_5$	20	5	3	23	3	2
eDEGPOA$_{0.8}$ vs	+	−	=	+	−	=
DE$_1$	15	2	11	19	1	8
DE$_2$	19	1	8	21	1	6
DE$_3$	16	2	10	17	1	10
DE$_4$	25	1	2	25	1	2
DE$_5$	23	1	4	26	1	1

"+" denotes wins; "−" denotes losses; "=" denotes ties.

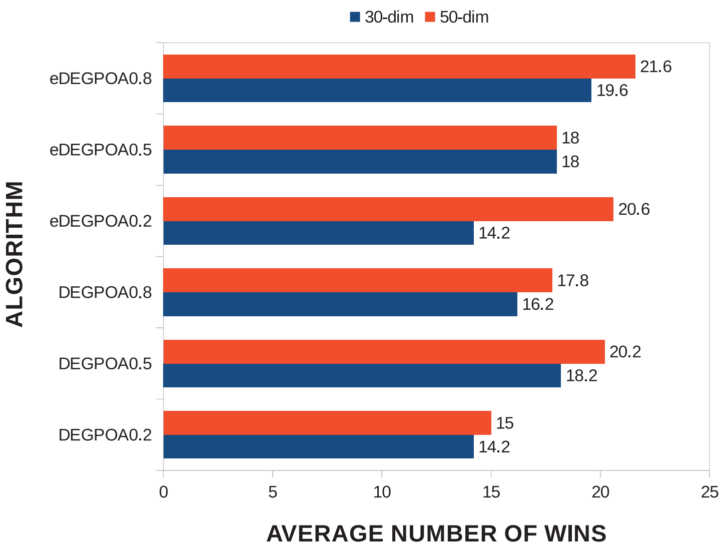


Fig. 2. Average number of wins of DEGPOA and eDEGPOA against the standard DE algorithms.

Table 4. Statistical comparisons between DEGPOA and other algorithms on the CEC-2013 test problems.

	Dimension					
	30			50		
DEGPOA$_{0.2}$ vs	+	−	=	+	−	=
SMADE	6	16	6	9	15	4
TLBSaDE	8	13	7	11	14	3
JANDE	6	18	4	9	17	2
DE_APC	11	12	5	12	12	4
TPC-GA	11	9	8	13	10	5
PVADE	10	14	4	13	13	2
CDASA	13	8	7	11	12	5
PLES	22	1	5	22	4	2
DEGPOA$_{0.5}$ vs	+	−	=	+	−	=
SMADE	7	15	6	9	15	4
TLBSaDE	9	12	7	9	13	6
JANDE	6	17	5	13	13	2
DE_APC	11	12	5	11	12	5
TPC-GA	11	9	8	13	9	6
PVADE	11	12	5	15	11	2
CDASA	14	8	6	14	11	3
PLES	23	1	4	21	2	5
DEGPOA$_{0.8}$ vs	+	−	=	+	−	=
SMADE	5	21	2	5	19	4
TLBSaDE	6	17	5	8	18	2
JANDE	0	21	7	4	16	8
DE_APC	10	15	3	10	15	3
TPC-GA	10	15	3	12	12	4
PVADE	8	15	5	10	14	4
CDASA	9	12	7	9	13	6
PLES	16	7	5	17	8	3

"+" denotes wins; "−" denotes losses; "=" denotes ties.

top-performing algorithms for the specific test suite, with their parameters being already tuned. This tuning procedure requires a significant amount of computational resources that can be even orders of magnitude higher than the one specified in the test suite. A fair comparison would suggest to add this extra computational budget also to our approaches, since they do not require any preprocessing. However, this budget is not reported in any relevant source. Thus, we were obligated to use only the standard computational budget specified by the test suite.

Even under this shortcoming, DEGPOA and eDEGPOA were highly competitive to the other approaches. Especially the (e)DEGPOA$_{0.2}$ and (e)DEGPOA$_{0.5}$ approaches were able to achieve same or higher number of wins than half or more of the rest of the algorithms, with better results being achieved in the higher

Table 5. Statistical comparisons between eDEGPOA and other algorithms on the CEC-2013 test problems.

	Dimension					
	30			50		
eDEGPOA$_{0.2}$ vs	+	−	=	+	−	=
SMADE	6	16	6	7	15	6
TLBSaDE	8	13	7	11	13	4
JANDE	5	18	5	8	16	4
DE_APC	11	12	5	12	12	4
TPC-GA	11	10	7	12	10	6
PVADE	10	15	3	11	13	4
CDASA	12	9	7	11	13	4
PLES	22	2	4	20	5	3
eDEGPOA$_{0.5}$ vs	+	−	=	+	−	=
SMADE	6	17	5	9	15	4
TLBSaDE	8	14	6	9	13	6
JANDE	6	18	4	12	13	3
DE_APC	11	11	6	11	12	5
TPC-GA	11	10	7	13	9	6
PVADE	11	14	3	15	12	1
CDASA	15	9	4	13	12	3
PLES	22	1	5	21	3	4
eDEGPOA$_{0.8}$ vs	+	−	=	+	−	=
SMADE	5	18	5	7	16	5
TLBSaDE	6	15	7	8	16	4
JANDE	4	19	5	7	17	4
DE_APC	10	13	5	11	13	4
TPC-GA	10	11	7	13	11	4
PVADE	11	12	5	12	10	6
CDASA	12	9	7	11	11	6
PLES	19	5	4	17	5	6

"+" denotes wins; "−" denotes losses; "=" denotes ties.

dimensional cases. Indeed, for the three DEGPOA algorithms the average numbers of wins are equal to 10.9, 11.5, and 8.0, in the 30-dimensional problems, while the corresponding numbers for the 50-dimensional case are 12.5, 13.1, and 9.4 (see Table 4). Similarly, eDEGPOA variants achieved 10.6, 11.2, and 9.6 wins on average in the 30-dimensional case, and 11.5, 11.6, and 10.8 wins on average for the 50-dimensional case (see Table 5). For completeness purposes, the average solution errors achieved by the best-performing DEGPOA$_{0.5}$ and eDEGPOA$_{0.5}$ approaches are reported in Tables 6–8 for both dimensions.

Finally, we compared the six variants of our approach with each other. In order to better visualize the results, we calculated the *rank* of each algorithm, defined as the difference between the total number of wins, w_{alg}, and the total number

Table 6. Average errors for the CEC-2013 test problems f_1-f_9.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
DEGPOA$_{0.5}$									
30	$0.00e+00$	$8.47e+06$	$2.49e+08$	$2.56e+04$	$0.00e+00$	$2.57e+01$	$1.00e+02$	$2.10e+01$	$2.65e+01$
50	$0.00e+00$	$1.89e+07$	$2.66e+09$	$5.58e+04$	$2.86e-07$	$4.66e+01$	$1.41e+02$	$2.12e+01$	$5.25e+01$
eDEGPOA$_{0.5}$									
30	$0.00e+00$	$1.20e+07$	$5.97e+08$	$3.00e+04$	$0.00e+00$	$2.48e+01$	$9.41e+01$	$2.10e+01$	$2.62e+01$
50	$0.00e+00$	$3.06e+07$	$3.86e+09$	$5.53e+04$	$4.22e-10$	$4.64e+01$	$1.41e+02$	$2.12e+01$	$5.29e+01$
SMADE									
30	$0.00e+00$	$0.00e+00$	$9.82e+03$	$0.00e+00$	$0.00e+00$	$2.67e+00$	$3.25e+01$	$2.10e+01$	$2.23e+01$
50	$0.00e+00$	$0.00e+00$	$3.81e+05$	$0.00e+00$	$0.00e+00$	$4.30e+01$	$4.32e+01$	$2.11e+01$	$4.36e+01$
TLBSaDE									
30	$0.00e+00$	$6.73e+03$	$3.91e+01$	$2.34e+00$	$0.00e+00$	$1.04e-02$	$1.54e+01$	$2.08e+01$	$2.69e+01$
50	$0.00e+00$	$1.68e+05$	$7.08e+05$	$6.58e+02$	$0.00e+00$	$4.07e+01$	$4.96e+01$	$2.11e+01$	$6.09e+01$
JANDE									
30	$0.00e+00$	$1.29e+05$	$9.84e+06$	$1.97e+04$	$1.26e-08$	$7.93e+00$	$9.82e+00$	$2.09e+01$	$2.10e+01$
50	$2.76e-08$	$6.05e+05$	$4.78e+07$	$8.34e+04$	$2.43e-06$	$4.30e+01$	$2.94e+01$	$2.11e+01$	$5.33e+01$
DE_APC									
30	$0.00e+00$	$1.75e+05$	$3.21e+06$	$2.20e-01$	$0.00e+00$	$9.35e+00$	$2.18e+01$	$2.09e+01$	$3.07e+01$
50	$0.00e+00$	$3.60e+05$	$6.98e+06$	$1.53e+00$	$0.00e+00$	$3.90e+01$	$3.66e+01$	$2.11e+01$	$6.09e+01$
TPC-GA									
30	$0.00e+00$	$2.44e+05$	$3.80e+07$	$1.38e+01$	$0.00e+00$	$2.43e+01$	$2.91e+01$	$2.10e+01$	$3.61e+01$
50	$0.00e+00$	$4.76e+05$	$1.06e+08$	$3.33e+00$	$0.00e+00$	$4.72e+01$	$4.17e+01$	$2.12e+01$	$7.30e+01$
PVADE									
30	$0.00e+00$	$2.12e+06$	$1.65e+03$	$1.70e+04$	$1.40e-07$	$8.29e+00$	$1.29e+00$	$2.09e+01$	$6.30e+00$
50	$0.00e+00$	$2.04e+05$	$7.48e+06$	$2.20e+02$	$1.39e-03$	$7.36e+01$	$2.07e+01$	$2.11e+01$	$2.60e+01$
CDASA									
30	$0.00e+00$	$9.52e+05$	$4.54e+07$	$1.83e-01$	$8.19e-06$	$3.54e+01$	$6.95e+01$	$2.09e+01$	$2.35e+01$
50	$0.00e+00$	$1.93e+06$	$2.18e+08$	$1.58e-02$	$8.40e-06$	$4.80e+01$	$1.04e+02$	$2.11e+01$	$4.67e+01$
PLES									
30	$0.00e+00$	$1.34e+07$	$1.94e+09$	$4.47e+04$	$0.00e+00$	$7.77e+01$	$1.18e+02$	$2.09e+01$	$3.31e+01$
50	$0.00e+00$	$1.59e+07$	$5.06e+09$	$5.43e+04$	$3.19e-10$	$9.70e+01$	$1.28e+02$	$2.11e+01$	$6.15e+01$

Table 7. Average errors for the CEC-2013 test problems f_{10}-f_{19}.

	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}
DEGPOA$_{0.5}$										
30	$1.79e+00$	$3.32e-01$	$8.90e+01$	$1.40e+02$	$1.80e+01$	$3.79e+03$	$1.33e+00$	$3.05e+01$	$1.41e+02$	$9.96e-01$
50	$6.39e+00$	$2.93e-01$	$2.70e+02$	$3.66e+02$	$1.15e+01$	$7.73e+03$	$1.79e+00$	$5.08e+01$	$3.25e+02$	$1.82e+00$
eDEGPOA$_{0.5}$										
30	$2.01e+00$	$7.80e-02$	$1.05e+02$	$1.57e+02$	$1.73e+01$	$3.79e+03$	$1.46e+00$	$3.05e+01$	$1.48e+02$	$9.63e-01$
50	$7.08e+00$	$2.15e-01$	$3.10e+02$	$4.03e+02$	$2.22e+01$	$7.90e+03$	$1.82e+00$	$5.09e+01$	$3.33e+02$	$1.80e+00$
SMADE										
30	$1.84e-02$	$1.09e+01$	$5.72e+01$	$1.28e+02$	$1.33e+02$	$4.10e+03$	$1.31e-01$	$3.48e+01$	$8.33e+01$	$2.55e+00$
50	$2.47e-02$	$4.81e+01$	$1.57e+02$	$3.35e+02$	$3.41e+02$	$8.54e+03$	$8.96e-02$	$6.57e+01$	$1.93e+02$	$5.43e+00$
TLBSaDE										
30	$1.62e-02$	$0.00e+00$	$4.99e+01$	$8.58e+01$	$3.07e+01$	$3.61e+03$	$1.48e+00$	$3.25e+01$	$7.68e+01$	$2.67e+00$
50	$1.76e-02$	$0.00e+00$	$1.20e+02$	$2.19e+02$	$8.25e+02$	$7.69e+03$	$1.80e+00$	$7.95e+01$	$1.81e+02$	$7.57e+00$
JANDE										
30	$7.91e-02$	$0.00e+00$	$4.28e+01$	$7.08e+01$	$1.33e+00$	$4.83e+03$	$2.28e+00$	$3.04e+01$	$1.23e+02$	$1.10e+00$
50	$1.47e-01$	$1.95e-02$	$9.72e+01$	$1.76e+02$	$8.01e+00$	$9.48e+03$	$3.13e+00$	$5.08e+01$	$2.18e+02$	$2.24e+00$
DE_APC										
30	$6.42e-02$	$3.08e+00$	$3.17e+01$	$7.55e+01$	$3.84e+03$	$4.14e+03$	$2.46e+00$	$5.92e+01$	$6.04e+01$	$2.30e+00$
50	$6.71e-02$	$3.44e+01$	$5.96e+01$	$1.55e+02$	$9.96e+03$	$9.34e+03$	$3.24e+00$	$1.72e+02$	$1.05e+02$	$5.08e+00$
TPC-GA										
30	$8.68e-02$	$2.39e+01$	$4.14e+01$	$8.41e+01$	$9.25e+02$	$3.97e+03$	$2.50e+00$	$5.44e+01$	$6.96e+01$	$3.28e+00$
50	$1.05e-01$	$5.57e+01$	$9.62e+01$	$1.92e+02$	$2.55e+03$	$9.40e+03$	$3.38e+00$	$1.15e+02$	$1.68e+02$	$8.92e+00$
PVADE										
30	$2.16e-02$	$5.84e+01$	$1.15e+02$	$1.31e+02$	$3.20e+03$	$5.61e+03$	$2.39e+00$	$1.02e+02$	$1.82e+02$	$5.40e+00$
50	$5.99e-01$	$1.68e+02$	$2.57e+02$	$3.06e+02$	$7.34e+03$	$1.25e+04$	$3.39e+00$	$2.38e+02$	$3.87e+02$	$2.12e+01$
CDASA										
30	$3.55e-02$	$1.17e+00$	$1.17e+02$	$1.86e+02$	$6.64e+02$	$3.87e+03$	$3.26e-01$	$3.40e+01$	$1.96e+02$	$2.10e+00$
50	$4.66e-02$	$2.15e+00$	$2.67e+02$	$4.11e+02$	$1.08e+03$	$7.33e+03$	$4.97e-01$	$5.82e+01$	$4.43e+02$	$3.69e+00$
PLES										
30	$1.18e+01$	$1.65e+02$	$2.15e+02$	$3.29e+02$	$2.61e+03$	$4.39e+03$	$1.32e+00$	$2.43e+02$	$2.57e+02$	$2.41e+01$
50	$2.99e+01$	$3.53e+02$	$4.41e+02$	$6.38e+02$	$5.06e+03$	$8.51e+03$	$2.07e+00$	$6.01e+02$	$6.33e+02$	$1.28e+02$

Table 8. Average errors for the CEC-2013 test problems f_{20}-f_{28}.

	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}	f_{27}	f_{28}
DEGPOA$_{0.5}$									
30	1.30e + 01	2.87e + 02	6.93e + 01	4.34e + 03	2.67e + 02	2.67e + 02	2.01e + 02	9.46e + 02	3.00e + 02
50	2.33e + 01	5.02e + 02	2.24e + 01	9.07e + 03	3.38e + 02	3.36e + 02	2.04e + 02	1.57e + 03	4.00e + 02
eDEGPOA$_{0.5}$									
30	1.30e + 01	2.92e + 02	1.01e + 02	4.66e + 03	2.69e + 02	2.69e + 02	2.01e + 02	8.31e + 02	3.00e + 02
50	2.34e + 01	5.39e + 02	4.99e + 01	9.17e + 03	3.39e + 02	3.39e + 02	2.04e + 02	1.67e + 03	4.00e + 02
SMADE									
30	1.05e + 01	3.27e + 02	1.79e + 02	4.22e + 03	2.32e + 02	2.78e + 02	2.15e + 02	6.47e + 02	3.88e + 02
50	1.92e + 01	8.46e + 02	3.39e + 02	9.89e + 03	3.00e + 02	3.68e + 02	2.91e + 02	1.18e + 03	1.07e + 03
TLBSaDE									
30	1.06e + 01	2.67e + 02	2.90e + 02	4.34e + 03	3.03e + 02	2.96e + 02	2.00e + 02	1.19e + 03	2.96e + 02
50	1.93e + 01	3.12e + 02	2.59e + 03	9.68e + 03	3.98e + 02	3.79e + 02	2.01e + 02	2.17e + 03	4.00e + 02
JANDE									
30	1.16e + 01	2.94e + 02	5.16e + 01	4.61e + 03	2.48e + 02	2.60e + 02	2.58e + 02	7.22e + 02	3.00e + 02
50	2.15e + 01	8.24e + 02	3.10e + 01	9.48e + 03	2.89e + 02	3.17e + 02	3.97e + 02	1.16e + 03	9.43e + 02
DE_APC									
30	1.26e + 01	2.67e + 02	4.56e + 03	4.18e + 03	2.92e + 02	2.99e + 02	3.28e + 02	1.19e + 03	3.00e + 02
50	2.23e + 01	6.81e + 02	1.06e + 04	9.09e + 03	3.84e + 02	3.83e + 02	4.09e + 02	2.14e + 03	6.97e + 02
TPC-GA									
30	1.37e + 01	2.92e + 02	1.27e + 03	4.33e + 03	2.74e + 02	2.98e + 02	3.25e + 02	1.03e + 03	3.00e + 02
50	2.34e + 01	7.93e + 02	3.51e + 03	9.93e + 03	3.77e + 02	3.86e + 02	4.22e + 02	2.03e + 03	4.59e + 02
PVADE									
30	1.13e + 01	3.19e + 02	2.50e + 03	5.81e + 03	2.02e + 02	2.30e + 02	2.18e + 02	3.26e + 02	3.00e + 02
50	2.07e + 01	9.65e + 02	7.72e + 03	1.18e + 04	2.78e + 02	3.54e + 02	3.47e + 02	1.11e + 03	4.62e + 02
CDASA									
30	1.48e + 01	2.77e + 02	4.89e + 02	5.41e + 03	2.98e + 02	3.15e + 02	2.91e + 02	1.08e + 03	3.87e + 02
50	2.43e + 01	6.86e + 02	7.32e + 02	1.01e + 04	3.74e + 02	4.04e + 02	3.44e + 02	1.60e + 03	1.04e + 03
PLES									
30	1.43e + 01	3.30e + 02	3.25e + 03	5.00e + 03	2.97e + 02	3.27e + 02	2.46e + 02	1.15e + 03	2.08e + 03
50	2.37e + 01	7.58e + 02	6.50e + 03	1.02e + 04	3.83e + 02	4.44e + 02	4.26e + 02	2.03e + 03	4.02e + 03

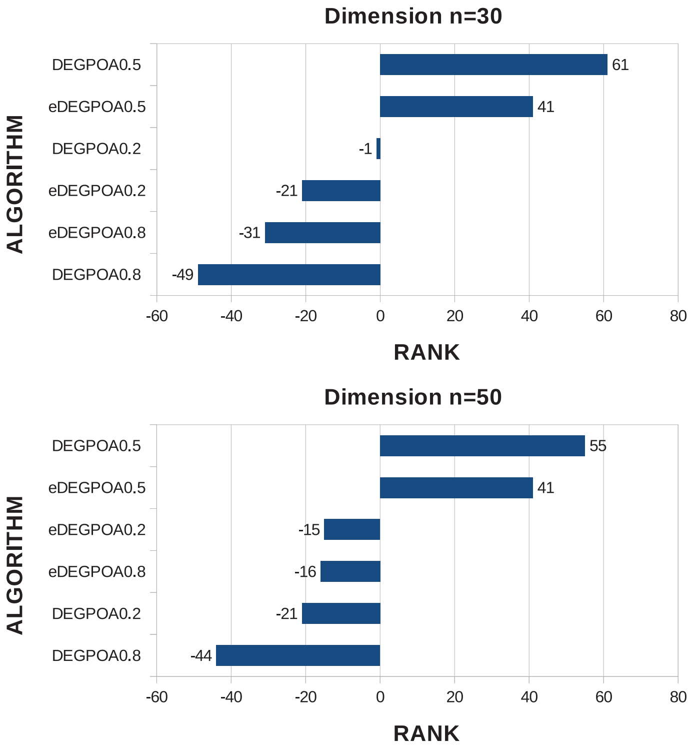


Fig. 3. Ranks of the proposed algorithms in comparisons among them for the 30-dimensional case (upper figure) and the 50-dimensional case (lower figure).

of losses, l_{alg}, it achieved in all comparisons, i.e.,

$$\texttt{rank}(\text{alg}) = w_{\text{alg}} - l_{\text{alg}} \,.$$

Since each algorithm is compared to five others on 28 test problems, it holds that

$$0 \leqslant w_{\text{alg}}, l_{\text{alg}} \leqslant 140 \,,$$

and, hence,

$$-140 \leqslant \texttt{rank}(\text{alg}) \leqslant 140 \,.$$

Figure 3 illustrates the ranks for all DEGPOA and eDEGPOA algorithms. As we can see, DEGPOA$_{0.5}$ and eDEGPOA$_{0.5}$ clearly dominated the rest. This is a direct consequence of their better initial parameter pair, $(\alpha, \beta) = (0.5, 0.5)$, which proved to be better than the other two (distant) initial parameter pairs. Note that the standard DE with these parameters exhibited inferior performance than all DEGPOA and eDEGPOA approaches. This implies that, even under defective initial parameters, the grid-based adaptation method is highly beneficial for the algorithm. On the other hand, when starting from a favorable parameter pair the proposed approach can significantly boost performance.

Summarizing our findings, although DEGPOA and eDEGPOA are more suitable for computationally demanding high-dimensional problems, they both exhibited satisfactory performance against some of the most competitive adaptive DE approaches such as SMADE, TLBSaDE, and JANDE. Moreover, all DEGPOA and eDEGPOA approaches proved to be competitive against other DE-based approaches such as DE_APC and PVADE, as well as against different algorithms such as TPC_GA, CDASA and PLES. This is a very promising result given that our approaches require only the computational budget specified by the test suite and no additional preprocessing or preliminary experimentation. Moreover, previous observations regarding the performance improvements as the dimension increases[32,33] were also verified for the CEC-2013 test suite. This is observed especially for the 50-dimensional test problems and regardless of the initial parameter pair.

5. Conclusions

Parameter tuning is a laborious and time-consuming task, intimately related to the corresponding optimization problem. However, it is a necessary procedure of vital importance for most algorithms when problems of high complexity are solved. This necessity has resulted in a number of adaptive algorithms, which are based mostly on ad-hoc procedures especially designed for the specific algorithm.

The present work offered an assessment of the Differential Evolution algorithm equipped with the recently proposed grid-based parameter adaptation method on the mainstream CEC-2013 test suite. The studied DEGPOA and eDEGPOA algorithms adopt a general-purpose grid-based search technique for adaptive control of their scalar parameters and mutation operator. The underlying parameter-control task is a mixed integer optimization problem itself. The resulting approaches do not require any preprocessing phase contrary to most of the competing algorithms.

The reported results verify the competitiveness of the two approaches. This comes despite the fact that the experimental setting of the CEC-2013 test suite is rather restrictive due to the limited computational budget, and the fact that the parameters of the competing algorithms were already tuned.

References

1. Complementary material: SOCO special issue on large scale continuous optimization problem. http://sci2s.ugr.es/EAMHCO#ComplementaryMaterial:SOCOSpecialIssueonLargeScaleContinuousOptimizationProblems.
2. 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.
3. T. Bartz-Beielstein, *Experimental Research in Evolutionary Computation: The New Experimentalism* (Springer, 2006).
4. H.-G. Beyer, *The Theory of Evolution Strategies* (Springer, Berlin, 2001).
5. M. Birattari, *Tuning Metaheuristics: A Machine Learning Perspective* (Springer, 2009).

6. S. Biswas, S. Kundu, S. Das and A. V. Vasilakos, Teaching and learning best differential evoltuion with self adaptation for real parameter optimization, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 1115–1122.
7. J. Brest, B. Bošković, A. Zamuda, I. Fister and E. Mezura-Montes, Real parameter single objective optimization using self-adaptive differential evolution algorithm with more strategies, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 377–383.
8. J. Brest, S. Greiner, B. Bošković, M. Mernik and V. Žumer, Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems, *IEEE Transactions on Evolutionary Computation* **10**(6) (2006) 646–657.
9. F. Caraffini, F. Neri, J. Cheng, G. Zhang, L. Picinali, G. Iacca and E. Mininno, Super-fit multicriteria adaptive differential evolution, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 1678–1685.
10. S. Chen, J. Montgomery and A. Bolufé-Röhler, Measuring the curse of dimensionality and its effects on particle swarm optimization and differential evolution, *Applied Intelligence* **42**(3) (April 2015) 514–526.
11. S. Das and P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Transactions on Evolutionary Computation* **15**(1) (2011) 4–31.
12. L. dos Santos Coelho, H. V. H. Ayala and R. Z. Freire, Population's variance-based adaptive differential evolution for real parameter optimization, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 1672–1677.
13. A. E. Eiben and S. K. Smit, Evolutionary algorithm parameters and methods to tune them, in *Autonomous Search*, eds. Y. Hamadi, E. Monfroy and F. Saubion (Springer, Berlin Heidelberg, 2011), Chapter 2, pp.'15–36.
14. A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing* (Springer-Verlag, 2015).
15. S. M. Elsayed, R. A. Sarker and D. L. Essam, A genetic algorithm for solving the CEC'2013 competition problems on real-parameter optimization, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 356–360.
16. S. M. M. Elsayed, R. A. Sarker and T. Ray, Differential evolution with automatic parameter configuration for solving the CEC2013 competition on real-parameter optimization, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 1932–1937.
17. M. Gendreau and J. Potvin, *Handbook of Metaheuristics*, 2nd edn. (Springer New York Dordrecht, Heidelberg London, 2010).
18. A. Gogna and A. Tayal, Metaheuristics: Review and application, *Journal of Experimental & Theoretical Artificial Intelligence* **25**(4) (2013) 503–526.
19. H. H. Hoos, Automated algorithm configuration and parameter tuning, in *Autonomous Search*, eds. Y. Hamadi, E. Monfroy and F. Saubion (Springer, Berlin Heidelberg, 2011), Chapter 3, pp. 37–72.
20. P. Korošec and J. Šilc, The continuous differential ant-stigmergy algorithm applied on real-parameter single objective optimization problems, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 1658–1663.
21. J. J. Liang, B. Y. Qu, P. N. Suganthan and A. G. Hernández-Díaz, Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report **201212**, 2013.
22. I. Loshchilov, T. Stuetzle and T. Liao, Ranking results of CEC'13 special session & competition on real-parameter single objective optimization (2013).

23. M. Lozano, F. Herrera and D. Molina, Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems, *Soft Computing* **15** (2011) 2085–2087.
24. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs* (Springer, Berlin, 1999).
25. G. Papa and J. Šilc, The parameter-less evolutionary search for real-parameter single objective optimization, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 1131–1137.
26. K. Parsopoulos and M. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications* (Information Science Publishing (IGI Global, 2010).
27. K. V. Price, R. M. Storn and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization* (Springer, Verlag, Berlin, 2005).
28. C. Segura, A. C. Coello Coello, E. Segredo and C. León, On the adaptation of the mutation scale factor in differential evolution, *Optimization Letters* **9**(1) (2015) 189–198.
29. R. Storn and K. Price, Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimization* **11** (1997) 341–359.
30. R. Tanabe and A. Fukunaga, Success-history based parameter adaptation for differential evolution, in *2013 IEEE Congress on Evolutionary Computation* (2013), pp. 71–78.
31. R. Tanabe and A. Fukunaga, Improving the search performance of SHADE using linear population size reduction, in *IEEE Congress on Evolutionary Computation* (2014).
32. V. A. Tatsis and K. E. Parsopoulos, Differential evolution with grid-based parameter adaptation, *Soft Computing* (2015) 1–23.
33. V. A. Tatsis and K. E. Parsopoulos, Grid search for operator and parameter control in differential evolution, in *Proc. of the 9th Hellenic Conf. on Artificial Intelligence* (*SETN '16*) (ACM, New York, USA, 2016), pp. 7:1–7:9.
34. V. A. Tatsis and K. E. Parsopoulos, Grid-based parameter adaptation in particle swarm optimization, in *Proc. of the 12th Metaheuristics Int. Conf.* (*MIC 2017*) (2017).
35. J. Torres-Jiménez and J. Pavón, Applications of metaheuristics in real-life problems, *Progress in Artificial Intelligence* **2**(4) (2014) 175–176.
36. J. Tvrdík, Competitive differential evolution, in *12th Int. Conf. on Soft Computing* (2006).
37. J. Tvrdík and R. Poláková, Competitive differential evolution applied to CEC 2013 problems, in *2013 IEEE Congress on Evolutionary Computation* (*CEC*) (IEEE, 2013), pp. 1651–1657.
38. M. Weber, V. Tirronen and F. Neri, Scale factor inheritance mechanism in distributed differential evolution, *Soft Computing* **14** (2010) 1187–1207.
39. D. Zaharie, A comparative analysis of crossover variants in differential evolution, *Proc. IMCSIT* (2007), pp. 171–181.
40. D. Zaharie, Influence of crossover on the behavior of differential evolution algorithms, *Applied Soft Computing* **9**(3) (2009) 1126–1138.
41. S. Z. Zhao, P. N. Suganthan and S. Das, Self-adaptive differential evolution with multi-trajectory search for large-scale optimization, *Soft Computing* **15**(11) (2011) 2175–2185.