Grid Search for Operator and Parameter Control in Differential Evolution

Vasileios A. Tatsis
Department of Computer Science & Engineering
University of Ioannina
GR-45110 Ioannina, Greece
vtatsis@cse.uoi.gr

Konstantinos E. Parsopoulos
Department of Computer Science & Engineering
University of Ioannina
GR-45110 Ioannina, Greece
kostasp@cse.uoi.gr

ABSTRACT

Evolutionary Algorithms constitute a very active research branch of Computational Intelligence. Typically, such algorithms are used for the detection of (sub-) optimal solutions in difficult optimization problems. Numerous works have provided experimental evidence of the remarkable efficiency of Evolutionary Algorithms. However, their performance has proved to be strongly connected to their proper parametrization. Various approaches have been proposed for (offline) tuning and (online) control of their parameters. Recently, a grid-based technique was proposed for parameter adaptation during the algorithm's run without user intervention, and it was validated on the Differential Evolution algorithm, which is widely known for its parameter sensitivity. Experimental results on high-dimensional test problems verified the effectiveness of the technique on controlling the scalar parameters and crossover type of the algorithm. The present work extends that study by considering another crucial component of the algorithm, namely the mutation operator type. Extensive experiments enrich and verify the previous evidence, suggesting that grid-based search can maintain competitive performance while absolving the user from the laborious parameter-tuning phase.

CCS Concepts

•**Computing methodologies** → **Bio-inspired approaches;**

1. INTRODUCTION

The complexity of real-world optimization problems has placed *metaheuristics* in a salient position among the available algorithmic artillery. Although solution optimality is often not guaranteed, metaheuristics can provide useful (sub-) optimal solutions in reasonable time, requiring only minimal knowledge of the problem. *Evolutionary Algorithms* (EAs) constitute an essential category of metaheuristics that model optimization procedures from nature [8].

EAs have dominated the relevant literature for decades due to their efficiency and effectiveness in solving hard optimization problems. However, their performance has proved to be intimately related to their parameterization [6]. Thus, the user is compelled to conduct a laborious preprocessing phase for the detection of promising parameter values and/or the adaptation of basic operators of the algorithm, on the specific problem at hand. A number of diverse techniques have been proposed in the literature for this purpose. They can be roughly categorized in two essential approaches, namely the offline tuning of parameters prior to the algorithm's execution, and the online, dynamic adaptation (control) of the parameters during its run [7].

Offline tuning is based on the detection of promising parameter values according to the outcome of a preprocessing phase that utilizes a set of test problems. Specifically, a set of parameter configurations are determined by the user and validated on all or a fraction of the test problems, usually under limited computational budget. The best-performing configurations are then selected to be applied to the problem of interest using all the available resources. The preprocessing phase may require even higher computational resources than the solution of the problem itself, although this is usually neglected in the overall algorithms' evaluations and comparisons. Sophisticated approaches such as Design of Experiments [2], F-Race [3], and ParamILS [10] are part of the state-of-the-art in this field.

In contrast to the offline approaches, online techniques modify the parameters by using feedback from the algorithm regarding its current performance. Thus, the parameters are adapted in real-time without the necessity of preprocessing or user intervention. This is a favorable property for online techniques, although their outcome is not re-usable in different problem instances or algorithms. Typical representatives of this category are the dynamic parameter adaptation approaches presented in [7, 10].

Differential Evolution (DE) has proved to be one of the most efficient and effective EAs. Due to its identified parameter sensitivity, a number of adaptive variants were proposed. Common approaches include SHADE [16] and L-SHADE [17] as well as the variants proposed in [4, 20, 21, 23, 24]. Multi-trajectory search was used in the self-adaptive scheme of [25], while distributed adaptation with inheritance and subpopulation connectivity was proposed in [22]. Further insight was offered in [14].

A grid-based adaptation technique was recently proposed in [19] and validated on DE. That approach is based on local search procedures applied on a grid constructed by the discretization of DE's parameter space. The search is driven by estimations of the algorithm's performance on neighboring parameter configurations on the grid. The experimental study of [19] was focused on the adaptation of the two scalar parameters of DE as well as its crossover type (binomial/exponential). The results revealed high competitiveness and superiority against other (DE-based and not) algorithms over an established test suite of high-dimensional test problems.

The present work extends the study of [19] by considering the mutation operator of the algorithm, along with its scalar parame-

SETN '16, May 18-20, 2016, Thessaloniki, Greece

DOI: http://dx.doi.org/10.1145/2903220.2903238

Algorithm 1 DIFFERENTIAL EVOLUTION

INPUT: N (population size); P (population); F, CR (scalar parameters); $T_{\max}$ (maximum number of iterations)
$t \leftarrow 0$
INITIALIZE $\left(P^{(t)}\right)$
while $(t < T_{max})$ **do**
  for $(i = 1 \ldots N)$ **do**
    $R_s^{(t)} \leftarrow$ RANDOM_INDICES $(I \setminus \{i\})$
    $u_i^{(t+1)} \leftarrow$ MUTATION $\left(R_s^{(t)}, x_i^{(t)}, x_g^{(t)}, F\right)$
    $v_i^{(t+1)} \leftarrow$ CROSSOVER $\left(x_i^{(t)}, u_i^{(t+1)}, CR\right)$
    EVALUATE $\left(v_i^{(t+1)}\right)$
    $x_i^{(t+1)} \leftarrow$ SELECTION $\left(x_i^{(t)}, v_i^{(t+1)}\right)$
  end for
  $P^{(t+1)} \leftarrow$ UPDATE $\left(P^{(t)}\right)$
  $t \leftarrow t+1$
  $g^{(t)} \leftarrow$ BEST_INDEX $\left(P^{(t)}\right)$
end while

ters. Thus, instead of tackling the crossover type between binomial and exponential, five different mutation operators are presented to the algorithm and adaptively selected during its run. For this purpose, the grid search is conducted over five grids instead of two in [19], thereby increasing the difficulty of finding proper parameter settings. We apply the proposed approach on the same test suite as in [19] and report comparisons with other algorithms following the same experimental setting. Statistical analysis is used to validate the results and derive useful conclusions.

The rest of the paper is organized as follows: Section 2 briefly presents DE and the grid-based search technique. The proposed approach is analyzed in Section 3. Experimental assessment is reported in Section 4. The paper closes with conclusions in Section 5.

2. BACKGROUND INFORMATION

The basic DE algorithm as well as the grid-based parameter adaptation technique are presented in the following paragraphs.

2.1 Differential Evolution

Differential Evolution (DE) is a widely recognized population-based optimization algorithm, originally introduced by Storn and Price [15]. DE utilizes a mutation procedure that involves the random selection of existing members of the population and their deterministic combination in order to produce new candidate solutions. Then, stochastic recombination takes place to produce new trial points that compete with the current members of the population. The simplicity and adaptability of DE has placed it among the most popular metaheuristics, also counting a significant number of enhanced variants [5].

Putting it formally, let

$$f : X \longrightarrow \mathbb{R}, \qquad X \subset \mathbb{R}^n,$$

be the objective function under consideration. DE employs a population,

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

of search points. Each point,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in X, \qquad i \in I \stackrel{\text{def}}{=} \{1, 2, \ldots, N\},$$

constitutes a candidate solution of the problem, and it is randomly and uniformly initialized in X.

At each iteration t, the population $P^{(t)}$ undergoes three main procedures, namely *mutation*, *crossover*, and *selection*, as oulined in Algorithm 1. Mutation combines two or more randomly selected members of $P^{(t)}$, producing a new vector $u_i^{(t+1)}$. There is a variety of mutation operators reported in literature. Among the most common ones are the following five operators that are used in the present study:

DE/Best/1:

$$u_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{1}$$

DE/Rand/1:

$$u_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)}\right), \tag{2}$$

DE/Current-to-Best:

$$u_i^{(t+1)} = x_i^{(t)} + F\left(x_g^{(t)} - x_i^{(t)} + x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{3}$$

DE/Best/2:

$$u_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)} + x_{r_3}^{(t)} - x_{r_4}^{(t)}\right), \tag{4}$$

DE/Rand/2:

$$u_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)} + x_{r_4}^{(t)} - x_{r_5}^{(t)}\right). \tag{5}$$

The index g denotes the best individual in $P^{(t)}$ with respect to its function value, i.e.,

$$g = \arg\min_{i \in I} \{f(x_i)\},$$

while $r_s \in I$ are randomly selected indices such that,

$$r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i.$$

Also, F is a fixed, user-defined parameter called *scale factor*, which usually assumes values in the range $(0, 1]$ [5].

Mutation is succeeded by the crossover procedure, where a *trial vector* v_i is produced for each x_i in the population. The most popular crossover operator is the *binomial* one,

$$v_{ij}^{(t+1)} = \begin{cases} u_{ij}^{(t+1)}, & \text{if } \mathscr{R} \leqslant CR \text{ or } j = \text{rand}(n), \\ x_{ij}^{(t)}, & \text{otherwise,} \end{cases} \tag{6}$$

where $j \in \{1, 2, \ldots, n\}$; $\mathscr{R}$ is a sample drawn from the uniform distribution over the range $[0, 1]$; $CR \in (0, 1]$ is another parameter of the algorithm called the *crossover rate*; and rand(n) is an integer randomly selected from the set $\{1, 2, \ldots, n\}$. The conditions of Eq. (6) ensure that the trial vector receives at least one component from $u_i^{(t+1)}$.

An alternative crossover operator is the *exponential* one. Under this scheme, $x_i^{(t)}$ is initially copied into $v_i^{(t+1)}$. Then, starting from a randomly selected index $k \in \{1, 2, \ldots, n\}$, the components $u_{ik}^{(t+1)}$ are copied in the corresponding components $v_{ik}^{(t+1)}$ as long as a stochastic condition is satisfied. When the condition fails, the

Algorithm 2 DEGPA

INITIALIZE(P)
$M \leftarrow 9$ /* *subpopulations* */
while (NOT TERMINATION) **do**
  /* *Dynamic's deployment phase* */
  EVOLVE(P, F, CR, t_p)
  /* *Performance estimation phase* */
  for ($i = 1 \ldots M$) **do**
    $P'_i \leftarrow P$
    ASSIGN(P'_i, F', CR') /* *use Eq. (8)* */
    EVOLVE(P'_i, F', CR', t_s)
  end for
  /* *Update primary population* */
  $P'_{best} \leftarrow BEST\_AOV(P'_1, \ldots, P'_9)$
  if $\left(\bar{f}_P - \bar{f}_{P'_{best}} \geqslant \varepsilon_{\min}\right)$ **then**
    $P \leftarrow P'_{best}$
    $(F, CR) \leftarrow (F'_{best}, CR'_{best})$
  end if
end while

procedure stops. Hence, the rest of the components $v_{ik}^{(t+1)}$ retain the values initially copied from $x_i^{(t)}$ [13].

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

$$x_i^{(t+1)} = \begin{cases} v_i^{(t+1)}, & \text{if } f\left(v_i^{(t+1)}\right) \leqslant f\left(x_i^{(t)}\right), \\ x_i^{(t)}, & \text{otherwise.} \end{cases} \quad (7)$$

The algorithm continues until a stopping criterion is satisfied. This criterion usually involves the maximum number of iterations or the desirable solution quality. When the algorithm terminates, the best detected solution x_g is reported.

2.2 Grid-Based Parameter Adaptation

In [19] an online adaptation technique for the parameters F and CR of the DE algorithm was proposed. The technique, called *DE with Grid-based Parameter Adaptation* (DEGPA) is based on a grid produced by discretizing the values of the two parameters in the range $(0,1]$, which is their most common domain [5]. Specifically, discretization step sizes λ_F and λ_{CR} are initially defined. The step sizes

$$\lambda_F = \lambda_{CR} = 0.1,$$

were adopted in [19], according to experimental evidence that associates smaller step sizes with marginal performance differences.

The discretization defines the following *grid*,

$$\mathscr{G} = \{(F, CR);\ F, CR, \in \{0.0, 0.1, \ldots, 1.0\}\},$$

which becomes the 2-dimensional parameter search space. Obviously, each point (F, CR) in the interior of $\mathscr{G}$ has 8 immediate neighboring points (F', CR'), corresponding to the search directions in the constructed grid. These points are defined as follows [19],

$$F' = F + a\,\lambda_F, \quad CR' = CR + b\,\lambda_{CR}, \quad a, b \in \{-1, 0, 1\}, \quad (8)$$

where $a = b = 0$ corresponds to the central pair (F, CR) itself.

The DEGPA algorithm, which is outlined in Algorithm 2, employs a *primary population*, which is assigned a selected parameter pair (F, CR). Initially, the parameters are set to the center of the grid, i.e., $(F, CR) = (0.5, 0.5)$. Given the parameters, the primary population is evolved for t_p iterations (for the moment let us assume that the mutation operator is known and fixed). This is called the *dynamic's deployment* phase of the algorithm [19]. According to suggestions in literature, the value,

$$t_p = 10 \times n,$$

where n is the problem's dimension, was proposed as a default fixed value in [19]. Alternatively, linearly increasing values were also considered.

After that, the resulting primary population is copied to $M = 9$ secondary populations (of same size as the primary one), each one assuming a different parameter pair (F', CR') according to Eq. (8). Let $P'_{a,b}$ denote the secondary population with the corresponding parameters for a and b in Eq. (8). Each secondary population is evolved for $t_e < t_p$ iterations and its performance is recorded. In order to minimize the computational cost of this step, t_e is selected to be much smaller than t_p. Typical values reported in [19] lie between 5 and 10 iterations.

Thus, a rough estimation of the performance trend is gained for each secondary population, i.e., for each alternative parameter pair (F', CR') neighboring with the current pair (F, CR). This second phase of the algorithm is called the *performance estimation* phase. Naturally, the secondary populations can be evolved either serially or in parallel [19].

Finally, a cycle of the algorithm is completed by selecting the best-performing secondary population and copying it to the primary one, along with its parameter pair. In order to fully exploit the findings of the unselected secondary populations, their best individuals are also sent to the new primary population, replacing its worst 8 individuals, if the latter have inferior objective values. From this point, a new cycle begins by initiating a dynamic's deployment phase anew. The algorithm is executed until a predefined stopping criterion is attained.

The secondary populations are compared with respect to their *average objective value* (AOV) [19] defined as,

$$\bar{f}_P = \frac{1}{N} \sum_{i=1}^{N} f(x_i), \qquad x_i \in P. \quad (9)$$

AOV was found to be less sensitive to occasional detection of steep local minima that may entrap the algorithm [19]. Additionally, in order to avoid marginal AOV improvements that can result in misleading adaptation of the parameter pair, a minimal improvement threshold $\varepsilon_{\min}$ is imposed for accepting a new parameter pair [19].

Hence, the best-performing secondary population $P'_{a,b}$ replaces the primary one, P, only if,

$$\bar{f}_P - \bar{f}_{P'_{a,b}} \geqslant \varepsilon_{\min} > 0.$$

Also, the maximum number of cycles, $c_{\max}$, of the algorithm can be estimated given the maximum number of iterations $T_{\max}$, as follows,

$$c_{\max} = \left\lfloor \frac{T_{\max}}{t_p + 9 t_e} \right\rfloor, \quad (10)$$

where $\lfloor . \rfloor$ defines the floor function. The DEGPA algorithm pro-

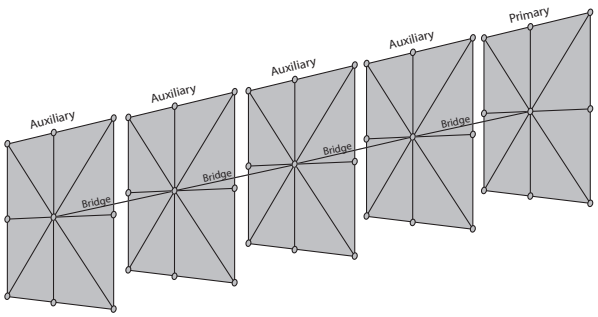

Figure 1: Bridging parallel grids through populations that correspond to different mutation operators.

duces a trajectory of parameter pairs by tracking the estimated improvement of performance in the parameter space $\mathscr{G}$.

In [19] DEGPA was accompanied with an enhanced version, namely eDEGPA. The eDEGPA approach closely follows the operation of DEGPA, although it introduces an additional secondary population that estimates the performance of the primary population's parameter pair under changes of the crossover type between exponential and binomial. This approach was found to produce even superior results than DEGPA for various test problems, motivating our study in the present work. Further details on DEGPA and eDEGPA can be found in [19].

3. THE PROPOSED APPROACH

The success of DEGPA and eDEGPA motivated our interest towards the further extension of the grid-based search technique. A key-factor in DE's operation and performance is the employed mutation operator, which is assumed to be fixed in [19]. This operator defines the dynamic of the algorithm with respect to the search directions that are used in order to sample the search space for new candidates solutions. For example, operators that involve the best individual, x_g, have been associated with rapid convergence but they are also more prone to be mislead by local minimizers. Contrary to this, operators with purely random selection of the involved vectors have been shown to promote diversity. Also, the use of one or two difference vectors can have remarkable impact on the algorithm's performance.

In the present work, we consider the main eDEGPA scheme, yet replacing the endeavor for controlling the crossover operator type in favor of the dynamic adaption of the mutation operator among the five operators defined in Eqs. (1)-(5). This requires the use of additional secondary populations besides the ones defined in the grid of the parameters F and CR. The additional populations inherit the primary population and its parameters, but assume different mutation operator, and they will be henceforth called the *bridge populations*. They can be conceived as bridging the current grid search with parallel grids corresponding to different mutation operators, as illustrated in Fig. 1.

We will henceforth call the proposed variant *DE with Grid-based Parameter and Operator Adaptation* (DEGPOA), and it is outlined in Algorithm 3. DEGPOA requires 9 secondary populations for the performance estimation under different parameter pairs (F,CR) as in DEGPA and eDEGPA, as well as 4 additional secondary populations that carry the same parameter pair as the primary one, but with different mutation operators. Thus, a total of 13 secondary populations are needed for the application of DEGPOA.

Similarly to DEGPA, the proposed approach is initialized with a primary population assuming initial parameters $(F,CR) = (0.5,0.5)$

Algorithm 3 DEGPOA

```
1: INITIALIZE(P)
2: M ← 13    /* subpopulations */
3: while (NOT TERMINATION) do
4:     /* Dynamic's deployment phase */
5:     EVOLVE(P, F, CR, OP_P, t_p)
6:     /* Performance estimation phase */
7:     for (i = 1...M) do
8:         P'_i ← P
9:         if (i ⩽ 9) then
10:            /* Normal secondary population */
11:            OP_{P'_i} ← OP_P
12:            ASSIGN(P'_i, F', CR', OP_{P'_i})    /* use Eq. (8) */
13:        else
14:            /* Bridging secondary population */
15:            CHANGE_OPERATOR(OP_{P'_i})
16:            (F', CR') ← (F, CR)
17:        end if
18:        EVOLVE(P'_i, F', CR', OP_{P'_i}, t_s)
19:    end for
20:    /* Update primary population */
21:    P'_best ← BEST_AOV(P'_1, ..., P'_13)
22:    if (f̄_P − f̄_{P'_best} ⩾ ε_min) then
23:        P ← P'_best
24:        (F, CR) ← (F'_best, CR'_best)
25:    end if
26: end while
```

and an initial mutation operator. The primary population is evolved for t_p iterations (dynamic's deployment phase). Then, it is copied to the 9 secondary populations of its own parameter grid, exactly as in the DEGPA approach described in the previous section. However, in DEGPOA the primary population is also copied in the 4 bridging populations, which assume same scalar parameters but different mutation operator than the primary population. For example, if the primary population has the initial parameters mentioned above and uses the DE/Rand/1 operator, then the bridging populations would assume the same scalar parameters $(F',CR') = (F,CR) = (0.5,0.5)$, but the DE/best/1, DE/Current-to-Best, DE/Rand/2, and DE/best/2 operators, respectively.

After that, each secondary population is evolved for t_e iterations (performance estimation phase). The best secondary population is selected to replace the primary population along with its parameters and mutation operator. This procedure can be viewed as jumping from one mutation operator's grid to another. The new primary population initiates a new cycle of the algorithm with a new dynamic's deployment phase and so on, as shown in Algorithm 3.

The selection of the primary population's initial mutation oper-

Table 1: Average errors and standard deviations for the proposed and base algorithms, for dimension $n = 50$ and 100.

Problem	DEGPOA		eDEGPOA		DE_{bin}		DE_{exp}		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
50-dimensional												
$F1$	$9.09e-14$	$3.67e-14$	$1.30e-13$	$3.86e-14$	$3.00e-17$	$7.69e-18$	$2.78e-17$	$6.29e-33$	$2.90e+02$	$5.69e+02$	$2.78e-17$	$6.29e-33$
$F2$	$6.25e+00$	$5.56e+00$	$7.06e+00$	$7.05e+00$	$3.87e+01$	$8.90e+00$	$3.31e-01$	$5.90e-02$	$7.72e+01$	$1.23e+01$	$7.69e-11$	$4.83e-11$
$F3$	$4.65e+01$	$2.12e+01$	$4.90e+01$	$1.97e+01$	$6.99e+01$	$3.58e+01$	$3.10e+01$	$8.65e+00$	$5.64e+07$	$1.42e+08$	$6.38e-01$	$1.49e+00$
$F4$	$4.78e-01$	$9.58e-01$	$1.99e-01$	$8.12e-01$	$3.21e+01$	$1.38e+01$	$4.79e-02$	$2.01e-01$	$1.12e+02$	$2.74e+01$	$3.72e+02$	$8.68e+01$
$F5$	$2.96e-04$	$1.48e-03$	$2.96e-04$	$1.48e-03$	$9.86e-04$	$2.76e-03$	$0.00e+00$	$0.00e+00$	$9.02e-01$	$1.82e+00$	$2.16e-01$	$5.64e-01$
$F6$	$1.49e-13$	$3.78e-14$	$2.58e-13$	$1.19e-13$	$7.16e-14$	$1.86e-14$	$1.39e-13$	$9.43e-15$	$3.23e+00$	$2.44e+00$	$1.90e+01$	$1.02e+00$
$F7$	$1.05e-15$	$3.21e-15$	$2.70e-11$	$1.33e-10$	$2.22e-15$	$1.17e-15$	$8.88e-17$	$1.96e-16$	$1.23e-09$	$1.45e-09$	$2.10e+01$	$1.38e+01$
$F8$	$1.64e+01$	$3.72e+01$	$1.55e+02$	$3.91e+02$	$9.02e+10$	$0.00e+00$	$9.02e+10$	$0.00e+00$	$9.02e+10$	$9.02e+06$	$9.03e+10$	$9.39e+07$
$F9$	$5.35e-02$	$9.50e-02$	$4.40e-03$	$8.08e-03$	$2.85e+02$	$5.30e+00$	$2.73e+02$	$7.40e-01$	$3.11e+02$	$4.98e+00$	$3.16e+02$	$7.03e+00$
$F10$	$9.67e-32$	$4.73e-31$	$1.78e-22$	$6.97e-22$	$1.53e+00$	$1.29e+00$	$6.50e-29$	$3.60e-29$	$7.72e+00$	$2.93e+00$	$9.25e+00$	$2.82e+00$
$F11$	$2.81e-02$	$5.00e-02$	$2.65e-02$	$1.01e-01$	$9.65e-01$	$2.02e+00$	$6.26e-05$	$1.30e-05$	$1.01e-02$	$1.26e-02$	$1.95e+02$	$3.65e+01$
$F12$	$8.34e-05$	$4.00e-04$	$1.27e-05$	$4.66e-05$	$5.82e+00$	$1.03e+01$	$5.26e-13$	$1.64e-13$	$8.23e+01$	$1.53e+02$	$1.14e+02$	$1.01e+01$
$F13$	$3.06e+01$	$1.99e+01$	$2.75e+01$	$5.77e+00$	$5.97e+01$	$2.22e+01$	$2.48e+01$	$1.31e+00$	$1.43e+07$	$3.29e+07$	$1.16e+02$	$1.43e+01$
$F14$	$2.79e-01$	$6.75e-01$	$1.99e-01$	$8.12e-01$	$3.35e+01$	$1.86e+01$	$3.55e-08$	$2.26e-08$	$6.76e+01$	$1.30e+01$	$2.71e+02$	$7.30e+01$
$F15$	$9.68e-15$	$3.43e-14$	$1.08e-12$	$3.78e-12$	$2.29e-01$	$6.07e-01$	$1.99e-24$	$3.22e-24$	$3.07e+00$	$5.32e+00$	$3.94e+01$	$1.25e+02$
$F16$	$3.77e-03$	$1.13e-02$	$8.22e-05$	$4.09e-04$	$5.64e+00$	$8.47e+00$	$1.56e-09$	$2.81e-10$	$5.60e+01$	$5.16e+01$	$2.23e+02$	$1.50e+01$
$F17$	$4.86e+00$	$6.32e+00$	$3.33e+00$	$4.82e+00$	$1.51e+01$	$1.43e+01$	$8.52e-01$	$4.92e-01$	$7.61e+06$	$2.44e+07$	$3.47e+02$	$2.18e+01$
$F18$	$8.59e-02$	$2.76e-01$	$4.39e-02$	$1.99e-01$	$5.73e+00$	$5.26e+00$	$1.28e-04$	$4.63e-05$	$6.76e+01$	$3.46e+01$	$3.59e+02$	$8.45e+01$
$F19$	$1.33e-23$	$5.99e-23$	$1.64e-16$	$7.87e-16$	$1.23e+00$	$9.26e-01$	$2.00e-24$	$1.50e-24$	$1.95e+02$	$5.01e+02$	$1.71e+03$	$5.84e+03$
100-dimensional												
$F1$	$2.43e-13$	$5.57e-14$	$3.34e-13$	$9.61e-14$	$1.12e-16$	$4.28e-17$	$7.77e-17$	$1.13e-17$	$4.67e+02$	$7.02e+02$	$5.55e-17$	$1.26e-32$
$F2$	$2.12e+01$	$9.36e+00$	$2.27e+01$	$8.64e+00$	$7.74e+01$	$7.77e+00$	$4.60e+00$	$4.24e-01$	$9.96e+01$	$1.16e+01$	$2.61e-03$	$1.30e-02$
$F3$	$1.10e+02$	$2.66e+01$	$1.03e+02$	$3.06e+01$	$4.43e+02$	$3.63e+02$	$8.01e+01$	$1.03e+01$	$1.52e+08$	$2.69e+08$	$1.23e+01$	$1.80e+01$
$F4$	$1.07e+00$	$2.07e+00$	$3.58e-01$	$1.25e+00$	$1.01e+02$	$2.25e+01$	$9.53e-03$	$4.76e-02$	$2.92e+02$	$5.16e+01$	$8.38e+02$	$1.39e+02$
$F5$	$2.96e-04$	$1.48e-03$	$1.65e-13$	$3.58e-14$	$2.93e-02$	$5.32e-02$	$2.55e-17$	$5.19e-18$	$5.95e+00$	$1.29e+01$	$2.68e+00$	$1.05e+01$
$F6$	$4.14e-13$	$9.95e-14$	$4.41e-12$	$1.27e-11$	$1.55e+00$	$3.88e-01$	$3.10e-13$	$1.62e-14$	$4.79e+00$	$1.87e+00$	$1.86e+01$	$2.45e+00$
$F7$	$2.26e-15$	$1.02e-14$	$2.94e-05$	$1.47e-04$	$1.39e-14$	$7.12e-15$	$3.80e-17$	$5.29e-17$	$8.67e-02$	$3.70e-01$	$6.35e+01$	$2.36e+01$
$F8$	$8.60e+02$	$2.18e+03$	$2.24e+03$	$2.93e+03$	$1.79e+11$	$0.00e+00$	$1.79e+11$	$0.00e+00$	$1.79e+11$	$1.92e+07$	$1.80e+11$	$3.54e+08$
$F9$	$4.80e-02$	$1.09e-01$	$2.04e-03$	$5.99e-03$	$5.43e+02$	$1.36e+01$	$5.06e+02$	$9.16e-01$	$5.87e+02$	$1.01e+01$	$6.08e+02$	$1.07e+01$
$F10$	$4.84e-28$	$2.40e-27$	$2.08e-23$	$6.17e-23$	$1.54e+01$	$3.31e+00$	$1.35e-28$	$3.86e-29$	$2.89e+01$	$1.01e+01$	$1.93e+01$	$5.10e+00$
$F11$	$7.29e-02$	$1.69e-01$	$4.44e-03$	$1.34e-02$	$4.31e+01$	$2.09e+01$	$1.25e-04$	$1.43e-05$	$2.80e+01$	$3.02e+01$	$4.82e+02$	$4.27e+01$
$F12$	$2.66e-03$	$9.35e-03$	$1.17e-03$	$3.72e-03$	$7.21e+01$	$3.21e+01$	$6.44e-11$	$1.52e-11$	$8.72e+02$	$2.55e+03$	$2.41e+02$	$1.23e+01$
$F13$	$6.74e+01$	$2.55e+01$	$6.30e+01$	$2.20e+01$	$2.76e+02$	$6.18e+01$	$6.13e+01$	$1.00e+00$	$9.37e+07$	$4.02e+08$	$2.59e+02$	$2.16e+01$
$F14$	$7.56e-01$	$2.12e+00$	$1.20e-01$	$3.30e-01$	$9.37e+01$	$1.56e+01$	$4.48e-02$	$2.24e-01$	$2.25e+02$	$4.59e+01$	$6.19e+02$	$9.25e+01$
$F15$	$6.36e-14$	$3.09e-13$	$7.16e-14$	$3.50e-13$	$3.67e+00$	$1.76e+00$	$7.10e-23$	$7.00e-23$	$5.99e+00$	$1.19e+01$	$5.57e+01$	$5.22e+01$
$F16$	$3.03e-03$	$9.53e-03$	$1.96e-03$	$9.43e-03$	$1.10e+02$	$3.80e+01$	$1.94e-02$	$9.70e-02$	$2.08e+02$	$1.49e+02$	$4.84e+02$	$2.08e+01$
$F17$	$1.92e+01$	$2.29e+01$	$1.30e+01$	$1.81e+01$	$1.78e+02$	$5.49e+01$	$1.19e+01$	$2.62e+00$	$4.36e+07$	$7.09e+07$	$7.04e+02$	$3.92e+01$
$F18$	$4.71e-01$	$1.01e+00$	$3.99e-02$	$1.99e-01$	$1.04e+02$	$4.39e+01$	$2.92e-04$	$6.77e-05$	$2.37e+02$	$7.02e+01$	$1.09e+03$	$4.15e+02$
$F19$	$5.54e-23$	$2.77e-22$	$5.86e-17$	$2.93e-16$	$1.17e+01$	$2.61e+00$	$4.79e-23$	$2.65e-23$	$4.70e+02$	$1.84e+03$	$5.83e+03$	$9.85e+03$

ator can be done in two ways. If an operator is known to perform well in the given problem, it is a reasonable choice to prefer it as the initial one. On the other hand, the initial operator can be randomly selected in absence of any relevant information.

Another significant issue that may require further investigation is the performance measure used for the assessment of the secondary populations. In [19], the AOV measure defined in Eq. (9) was preferred against the common choice of the overall best value, because it is less sensitive in temporary performance improvements that may be caused by the rapid detection of local minimizers (especially from the greedier operators). However, AOV is still based on a single performance aspect, namely the objective value, neglecting other aspects such as population diversity.

In order to probe the diversity properties of DEGPOA, we considered also a second, diversity-based performance measure, namely the *objective value standard deviation* (OVSD), defined as,

$$\sigma_{f_P} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(f(x_i) - \bar{f}_P\right)}, \qquad x_i \in P, \tag{11}$$

where $\bar{f}_P$ is the AOV defined in Eq. (9). OVSD measures the diversity of the objective function values of the population and can be used as a rapidly computed indicator of the population's diversity. Higher values of OVSD can be associated to higher diversity, which is preferable for alleviating premature convergence.

The concurrent use of two performance measure is rather simple and draws ideas from the concept of Pareto dominance in multi-objective optimization. Specifically, after the execution of t_e iterations by all secondary populations (including the bridging populations), their (AOV, OVSD) pairs, defined as,

$$(\bar{f}_{P_i'}, \sigma_{f_{P_i'}}),$$

are recorded and disposed in an external archive. Then, the non-dominated pairs, in terms of Pareto dominance, are identified. The non-dominated pairs are incomparable among them, since they are superior according to the one performance criterion but inferior according to the other. The final selection can be made either randomly among the non-dominated pairs or with respect to the overall best value of the corresponding secondary populations. The re-

Table 2: Average errors and standard deviations for the proposed and base algorithms, for dimension $n = 200$ and 500.

Problem	DEGPOA		eDEGPOA		DE_{bin}		DE_{exp}		CHC		GCMAES	
	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD	Mean	StD
200-dimensional												
$F1$	$5.82e-13$	$1.41e-13$	$7.12e-13$	$1.20e-13$	$6.39e-16$	$5.32e-16$	$1.78e-16$	$1.60e-17$	$9.61e+02$	$1.65e+03$	$1.17e-16$	$1.13e-17$
$F2$	$4.65e+01$	$9.30e+00$	$5.24e+01$	$1.21e+01$	$1.01e+02$	$5.90e+00$	$1.89e+01$	$1.05e+00$	$1.17e+02$	$7.60e+00$	$7.47e-02$	$2.44e-01$
$F3$	$2.04e+02$	$3.48e+01$	$2.12e+02$	$3.38e+01$	$6.38e+02$	$3.80e+02$	$1.79e+02$	$8.89e+00$	$2.54e+08$	$3.97e+08$	$1.24e+02$	$8.85e+01$
$F4$	$1.43e+00$	$3.11e+00$	$3.98e-01$	$1.32e+00$	$4.21e+02$	$5.63e+01$	$8.52e-02$	$3.98e-01$	$6.32e+02$	$8.43e+01$	$1.57e+03$	$1.54e+02$
$F5$	$4.45e-12$	$2.07e-11$	$3.57e-13$	$6.72e-14$	$3.00e-01$	$7.73e-01$	$7.49e-17$	$6.94e-18$	$1.02e+01$	$1.59e+01$	$1.13e+00$	$2.84e+00$
$F6$	$9.15e-13$	$2.21e-13$	$1.85e-12$	$1.76e-12$	$5.28e+00$	$9.65e-01$	$6.46e-13$	$2.53e-14$	$8.14e+00$	$2.26e+00$	$1.93e+01$	$7.39e-01$
$F7$	$6.34e-15$	$1.15e-14$	$6.41e-12$	$3.16e-11$	$1.78e-11$	$4.65e-11$	$2.25e-16$	$1.92e-16$	$3.95e-01$	$1.21e+00$	$1.25e+02$	$1.92e+01$
$F8$	$5.92e+03$	$7.34e+03$	$1.56e+04$	$1.62e+04$	$8.33e+11$	$0.00e+00$	$8.33e+11$	$0.00e+00$	$8.33e+11$	$3.09e+08$	$8.56e+11$	$3.36e+09$
$F9$	$3.79e-02$	$8.17e-02$	$5.54e-03$	$1.55e-02$	$1.13e+03$	$1.87e+01$	$1.01e+03$	$1.30e+00$	$1.18e+03$	$8.29e+00$	$1.22e+03$	$1.79e+01$
$F10$	$8.23e-28$	$4.11e-27$	$2.03e-21$	$1.02e-20$	$5.52e+01$	$1.02e+01$	$2.77e-28$	$5.31e-29$	$7.34e+01$	$6.25e+01$	$3.76e+01$	$2.78e+01$
$F11$	$5.20e-02$	$1.15e-01$	$3.68e-03$	$1.00e-02$	$3.95e+02$	$6.11e+01$	$2.55e-04$	$3.20e-05$	$4.03e+02$	$8.45e+01$	$1.08e+03$	$8.25e+01$
$F12$	$1.51e-03$	$4.22e-03$	$1.46e-03$	$7.27e-03$	$2.84e+02$	$4.97e+01$	$9.97e-10$	$2.01e-10$	$8.11e+02$	$1.58e+03$	$5.87e+02$	$4.52e+02$
$F13$	$1.47e+02$	$3.46e+01$	$1.37e+02$	$3.81e+01$	$7.52e+02$	$2.71e+02$	$1.40e+02$	$1.26e+01$	$2.06e+08$	$3.51e+08$	$5.92e+02$	$1.08e+02$
$F14$	$5.17e-01$	$1.41e+00$	$1.19e-01$	$5.97e-01$	$3.11e+02$	$3.66e+01$	$8.08e-03$	$4.04e-02$	$4.90e+02$	$5.23e+01$	$1.26e+03$	$1.81e+02$
$F15$	$7.95e-14$	$2.05e-13$	$1.29e-13$	$4.57e-13$	$1.17e+01$	$2.85e+00$	$3.71e-24$	$2.32e-24$	$1.40e+01$	$9.80e+00$	$1.95e+02$	$1.66e+02$
$F16$	$1.62e-02$	$4.13e-02$	$2.25e-05$	$6.82e-05$	$5.58e+02$	$7.96e+01$	$7.85e-09$	$1.11e-09$	$6.77e+02$	$6.04e+02$	$9.56e+02$	$3.33e+01$
$F17$	$5.99e+01$	$3.05e+01$	$4.71e+01$	$2.84e+01$	$1.03e+03$	$1.11e+02$	$3.71e+01$	$8.30e-01$	$1.17e+07$	$1.70e+07$	$1.49e+03$	$8.00e+01$
$F18$	$5.48e-02$	$2.06e-01$	$4.00e-02$	$1.99e-01$	$7.53e+02$	$6.55e+01$	$5.10e-04$	$9.97e-05$	$7.67e+02$	$2.14e+02$	$3.94e+03$	$3.91e+03$
$F19$	$2.43e-16$	$1.03e-15$	$1.72e-15$	$6.24e-15$	$4.04e+01$	$7.71e+00$	$1.67e-22$	$7.58e-23$	$7.51e+02$	$1.76e+03$	$2.53e+04$	$2.45e+04$
500-dimensional												
$F1$	$1.57e-12$	$2.76e-13$	$1.61e-12$	$4.01e-13$	$3.88e-05$	$7.93e-05$	$5.17e-16$	$1.36e-17$	$9.25e+02$	$1.27e+03$	n/a	n/a
$F2$	$8.93e+01$	$1.20e+01$	$8.76e+01$	$1.17e+01$	$1.25e+02$	$5.37e+00$	$5.38e+01$	$1.21e+00$	$1.35e+02$	$5.52e+00$	n/a	n/a
$F3$	$5.28e+02$	$8.52e+01$	$5.00e+02$	$2.83e+01$	$3.44e+04$	$1.62e+05$	$4.74e+02$	$1.48e+00$	$6.93e+08$	$1.72e+09$	n/a	n/a
$F4$	$1.87e+00$	$5.67e+00$	$3.98e-02$	$1.99e-01$	$2.35e+03$	$1.60e+02$	$7.12e-01$	$9.64e-01$	$2.11e+03$	$1.66e+02$	n/a	n/a
$F5$	$8.48e-13$	$1.51e-13$	$8.61e-13$	$1.54e-13$	$3.11e-01$	$5.07e-01$	$2.38e-16$	$1.18e-17$	$1.45e+01$	$2.52e+01$	n/a	n/a
$F6$	$3.28e-12$	$1.39e-12$	$2.91e-12$	$3.78e-13$	$1.49e+01$	$8.38e-01$	$1.64e-12$	$4.85e-14$	$1.27e+01$	$1.26e+00$	n/a	n/a
$F7$	$6.21e-13$	$2.14e-12$	$2.74e-13$	$1.36e-12$	$2.74e-03$	$6.49e-03$	$7.29e-16$	$3.58e-16$	$3.33e-05$	$1.12e-04$	n/a	n/a
$F8$	$9.60e+04$	$9.04e+04$	$1.11e+05$	$1.18e+05$	$4.94e+12$	$0.00e+00$	$4.94e+12$	$0.00e+00$	$4.94e+12$	$1.42e+08$	n/a	n/a
$F9$	$5.47e-02$	$1.37e-01$	$3.58e-03$	$9.41e-03$	$2.97e+03$	$3.17e+01$	$2.52e+03$	$2.10e+00$	$3.00e+03$	$1.64e+01$	n/a	n/a
$F10$	$1.41e-31$	$4.85e-31$	$1.51e-30$	$3.54e-30$	$1.36e+02$	$2.08e+01$	$9.79e-28$	$1.43e-28$	$1.64e+02$	$5.62e+01$	n/a	n/a
$F11$	$1.31e-01$	$3.27e-01$	$3.88e-03$	$9.22e-03$	$2.34e+03$	$9.22e+01$	$6.78e-04$	$3.60e-05$	$1.67e+03$	$1.44e+02$	n/a	n/a
$F12$	$1.55e-02$	$4.42e-02$	$1.25e-05$	$4.65e-05$	$1.02e+03$	$6.68e+01$	$6.80e-09$	$8.58e-10$	$1.62e+03$	$1.83e+03$	n/a	n/a
$F13$	$3.91e+02$	$5.28e+01$	$3.54e+02$	$3.01e+01$	$2.49e+03$	$3.13e+02$	$3.60e+02$	$9.23e+00$	$3.41e+08$	$4.29e+08$	n/a	n/a
$F14$	$7.57e-01$	$2.73e+00$	$1.23e-04$	$6.13e-04$	$1.67e+03$	$1.51e+02$	$3.93e-01$	$1.05e+00$	$1.59e+03$	$1.57e+02$	n/a	n/a
$F15$	$9.20e-13$	$3.33e-12$	$8.21e-13$	$3.77e-12$	$4.44e+01$	$5.59e+00$	$2.93e-18$	$7.16e-18$	$3.50e+01$	$1.20e+01$	n/a	n/a
$F16$	$1.04e-02$	$4.78e-02$	$2.74e-06$	$7.42e-06$	$2.02e+03$	$8.60e+01$	$2.05e-08$	$1.64e-09$	$1.92e+03$	$1.44e+03$	n/a	n/a
$F17$	$1.41e+02$	$4.51e+01$	$1.39e+02$	$3.57e+01$	$3.83e+03$	$1.41e+02$	$1.12e+02$	$1.02e+00$	$6.64e+08$	$1.64e+09$	n/a	n/a
$F18$	$3.24e-01$	$8.12e-01$	$4.02e-02$	$1.99e-01$	$3.37e+03$	$4.34e+02$	$1.25e-03$	$1.87e-04$	$2.74e+03$	$3.59e+02$	n/a	n/a
$F19$	$7.64e-16$	$2.69e-15$	$2.29e-21$	$1.13e-20$	$1.29e+02$	$2.34e+01$	$3.35e-21$	$2.15e-21$	$2.05e+03$	$4.03e+03$	n/a	n/a

quired time for this procedure is negligible since there are only 13 pairs in the archive. The DEGPOA variant with the two performance measures will be henceforth denoted as eDEGPOA.

4. EXPERIMENTAL ASSESSMENT

Following the experimental setting of [19], the test suite proposed in the *Special Issue on Large Scale Continuous Optimization Problems* of the *Soft Computing* journal [12] was used to validate the proposed DEGPOA and eDEGPOA approaches. The test suite consists of 19 scalable test functions, henceforth denoted as $F1-F19$. These include CEC 2008 test problems [18], as well as shifted and hybrid composition functions [12].

Three base algorithms are proposed in the test suite for comparisons, namely DE with exponential crossover, CHC, and GCMAES [1, 9, 12]. For the sake of completeness, we included also the DE with binomial crossover using the settings and source codes provided in [11]. The control parameters of the base DE approaches were set to their claimed optimal values $(F,CR) = (0.7, 0.5)$, along with the reportedly best-performing DE/Rand/1 mutation operator. In addition, the average performance for a number of different metaheuristics is provided in the test suite for comparison purposes [12].

The specific test suite was selected against others due to its problem diversity and high-dimensionality. All test problems were considered for dimension,

$$n = 50,\ 100,\ 200,\ 500.$$

For each test problem and dimension, both DEGPOA and eDEGPOA were applied assuming a computational budget of,

$$T_{\max} = 5000 \times n,$$

function evaluations, according to the typical setting of the specific test suite [12]. The competent algorithms require a preliminary parameter-tuning phase that is typically not accounted in their computational cost, while our approaches do not require this phase because of the online parameter tunning. Thus, it would be more fair to provide to our approaches additional running time. However, we restricted our budget to the strict requirements of the test suite in order to evaluate their performance competitiveness under tight resource limits.

Table 3: Number of wins (+), losses (−), and draws (=) of DEGPOA and eDEGPOA against the base algorithms.

		DEGPOA			eDEGPOA		
Dimension	Algorithm	+	−	=	+	−	=
50	DE_{bin}	14	3	2	13	3	3
	DE_{exp}	7	5	7	4	6	9
	CHC	18	0	1	19	0	0
	GCMAES	16	3	0	16	3	0
100	DE_{bin}	17	1	1	17	1	1
	DE_{exp}	7	5	7	5	7	7
	CHC	19	0	0	19	0	0
	GCMAES	16	3	0	16	3	0
200	DE_{bin}	17	1	1	17	1	1
	DE_{exp}	8	6	5	8	5	6
	CHC	19	0	0	19	0	0
	GCMAES	16	3	0	14	3	2
500	DE_{bin}	19	0	0	19	0	0
	DE_{exp}	8	7	4	11	6	2
	CHC	19	0	0	19	0	0
	GCMAES	n/a	n/a	n/a	n/a	n/a	n/a

The quality of the final solution was measured with the error from the optimal solution value,

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

where x^* is the solution achieved by the algorithm and x_{opt} is the known optimal solution of the problem. Since all algorithms are stochastic in nature, 25 independent experiments were conducted in order to approximate their performance distributions.

Following the setting of the test suite's DE-based approaches, DEGPOA and eDEGPOA assumed a fixed population size,

$$N = 60.$$

The initial parameter pair in all runs was the center of the grid,

$$(F, CR) = (0.5, 0.5).$$

Both approaches were considered with their exponential crossover, which was proved to be the best one also in [12, 19]. Regarding the rest of the parameters, the values,

$$t_e = 10, \qquad \varepsilon_{\min} = 10^{-2},$$

were used while t_p was linearly adapted between its minimum and maximum values,

$$t_p^{\min} = 10 \times n, \qquad t_p^{\max} = 14 \times n,$$

according to the suggestions in [19].

The experimental assessment consisted of two phases. In the first phase, full statistical comparisons between DEGPOA, eDEGPOA, and the base algorithms of the test suite were conducted. In the second phase, comparisons of average performance were made against the rest of the test suite's metaheuristics. This experimental practice is dictated by the specific test suite [12].

The base algorithms were applied using the source codes and settings provided in the test suite. The obtained solution errors were recorder for all algorithms and test problems. The mean and standard deviations are reported in Tables 1 and 2. For the GCMAES approach, 500-dimensional results were not attainable due to the excessive computation time required by the algorithm (this is reported as "n/a" in the Tables).

The results clearly show that both DEGPOA and eDEGPOA are very competitive against the base algorithms. They outperform DE_{bin}, CHC, and GCMAES in almost all test problems and dimensions. Also, they exhibit remarkable competitiveness against DE_{exp}, which was recognized as the best-performing base algorithm in the test suite [12]. We shall emphasize that the competitiveness of our approaches comes along with the depletion of the computational burden for parameter tuning, while for the rest of the approaches their most promising versions are used after additional parameter-tuning cost. This fact augments the usefulness of DEGPOA and eDEGPOA.

In order to statistically verify the observed performance differences with the base algorithms, pairwise Wilcoxon ranksum tests of the algorithms at confidence level 95% were conducted for all test functions. Each favorable comparison was counted as a win for the algorithm and denoted as "+". Negative comparisons were counted as losses and denoted as "−", while draws were denoted as "=". Table 3 summarizes the number of wins, losses, and draws of DEGPOA and eDEGPOA against the base algorithms.

An interesting observation is that, as dimension increases, both DEGPOA and eDEGPOA exhibit higher numbers of wins against the strongest competitor algorithm, namely DE_{exp}. Moreover, we can observe that the diversity-promoting eDEGPOA variant has marginal difference than DEGPOA, although it improves its performance with the problem's dimension. This is an indication that the online adaptation mechanism of DEGPOA is capable of preserving the necessary diversity even without the use of specialized mechanisms or diversity-oriented performance measures.

The second phase of experimental analysis included comparisons of expected performance against the rest of the test suite's algorithms [11, 12]. The comparisons were based on the algo-

Table 4: Number of problems where DEGPOA and eDEGPOA exhibited inferior and non-inferior average solution values against the rest of the algorithms.

	Non-Inferior (Dim.)				Inferior (Dim.)			
	50	100	200	500	50	100	200	500
DEGPOA								
EvoPROpt	17	17	17	16	2	2	2	3
EM323	9	9	10	12	10	10	9	7
SOUPDE	6	6	7	7	13	13	12	12
DE-D^{40}+M^m	6	7	8	8	13	12	11	11
GODE	6	6	7	7	13	13	12	12
MA-SSW-Chains	11	13	16	16	8	6	3	3
GaDE	7	7	8	7	12	12	11	12
RPSO-vm	13	13	16	16	6	6	3	3
jDElscop	6	6	7	7	13	13	12	12
SaDE-MMTS	7	7	9	11	12	12	10	8
MOS	7	6	7	7	12	13	12	12
Tuned IPSOLS	11	10	12	12	8	9	7	7
VXQR1	12	13	15	15	7	6	4	4
eDEGPOA								
EvoPROpt	17	17	16	17	2	2	3	2
EM323	9	10	12	16	10	9	7	3
SOUPDE	6	6	8	9	13	13	11	10
DE-D^{40}+M^m	7	8	9	11	12	11	10	8
GODE	6	7	8	9	13	12	11	10
MA-SSW-Chains	11	13	16	16	8	6	3	3
GaDE	7	7	8	7	12	12	11	12
RPSO-vm	14	14	16	16	5	5	3	3
jDElscop	6	6	7	9	13	13	12	10
SaDE-MMTS	7	8	9	12	12	11	10	7
MOS	7	7	7	7	12	12	12	12
Tuned IPSOLS	11	9	12	12	8	10	7	7
VXQR1	13	14	15	15	6	5	4	4

rithms' average error values provided in their original sources [11]. Table 4 reports the number of test problems were DEGPOA and eDEGPOA exhibited non-inferior or inferior average error values from the rest of the algorithms. Again, it is confirmed that both algorithms have similar non-inferior performance. However, as dimension increases, eDEGPOA has marginally better performance. Nevertheless, it worths noting that both algorithms outperform also non-DE-based algorithms, such as EvoPROpt, MA-SSW-Chains, RPRSO-vm, Tuned IPSOLS and VXQR1, in all dimensions.

5. CONCLUSIONS

The parameter tuning of metaheuristics is a laborious task that often consumes significant amount of time and computational resources, while the result is usually problem-dependent. The present work proposed an extension of a recently introduced grid-based search technique for dynamic parameter adaptation.

The technique was previously evaluated on the DE algorithm for the online control of its scalar parameters and crossover type. We extended this approach by considering another crucial aspect of DE's performance, namely the mutation operator. Moreover, we introduced also a diversity-promoting evaluation measure for the performance of the involved secondary populations.

The proposed approaches were evaluated on an established high-dimensional test suite, following the setting of previous studies. The obtained results revealed that the proposed adaptive approaches can release the user from the burden of parameter tuning, while attaining very competitive performance in all test problems using the same computational resources as the specially tuned algorithms.

The parameter grid search approach has opened various directions for further inquiry. More refined versions of the algorithm that are based on the literature of deterministic gradient-based optimization are under development. Furthermore, the grid-based adaptation technique is not explicitly dependent from the specific algorithm. Thus, its effectiveness on different algorithms will provide

additional insight.

6. REFERENCES

[1] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Proc. of the 2005 IEEE Congress on Evolutionary Computation*, pages 769–1776, 2005.

[2] T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation: The New Experimentalism*. Springer, 2006.

[3] M. Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*. Springer, 2009.

[4] 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):646–657, 2006.

[5] S. Das and P. N. Suganthan. Differential Evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.

[6] A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[7] A. E. Eiben and S. K. Smit. Evolutionary algorithm parameters and methods to tune them. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous Search*, chapter 2, pages 15–36. Springer, Berlin Heidelberg, 2011.

[8] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2015.

[9] Eshelman, L.J., and S. J.D. Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms*, 2:187–202, 1993.

[10] H. H. Hoos. Automated algorithm configuration and parameter tuning. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous Search*, chapter 3, pages 37–72. Springer, Berlin Heidelberg, 2011.

[11] M. Lozano, F. Herrera, and D. Molina. Evolutionary algorithms and other metaheuristics for continuous optimization problems.

[12] M. Lozano, F. Herrera, and D. Molina. Editorial: scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 15:2085–2087, 2011.

[13] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Verlag, Berlin, 2005.

[14] C. Segura, C. A. C. Coello, E. Segredo, and C. León. On the adaptation of the mutation scale factor in differential evolution. *Optimization Letters*, 9(1):189–198, 2015.

[15] R. Storn and K. Price. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization*, 11:341–359, 1997.

[16] R. Tanabe and A. Fukunaga. Success-history based parameter adaptation for differential evolution. In *IEEE Congress on Evolutionary Computation*, 2013.

[17] R. Tanabe and A. Fukunaga. Improving the search performance of SHADE using linear population size reduction. In *IEEE Congress on Evolutionary Computation*, 2014.

[18] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang. Benchmark functions for the cec 2008 special session and competition on large scale global optimization. *Nature Inspired Computation and Applications Laboratory, USTC, China*, pages 153–177, 2007.

[19] V. A. Tatsis and K. E. Parsopoulos. Differential evolution with grid-based parameter adaptation. *Soft Computing*, 2015, in press.

[20] J. Tvrdík. Competitive differential evolution. In *12th International Coference on Soft Computing*, 2006.

[21] J. Tvrdík and R. Poláková. Competitive differential evolution applied to CEC 2013 problems. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1651–1657. IEEE, 2013.

[22] M. Weber, V. Tirronen, and F. Neri. Scale factor inheritance mechanism in distributed differential evolution. *Soft Computing*, 14:1187–1207, 2010.

[23] D. Zaharie. A comparative analysis of crossover variants in differential evolution. *Proceedings of IMCSIT*, pages 171–181, 2007.

[24] D. Zaharie. Influence of crossover on the behavior of differential evolution algorithms. *Applied Soft Computing*, 9(3):1126–1138, 2009.

[25] S. Zhao, P. Suganthan, and S. Das. Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Comput.*, 15(11):2175–2185, 2011.