Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Empirical evaluation of distributed Differential Evolution on standard benchmarks

Javier Apolloni [a,*], José García-Nieto [b], Enrique Alba [b,c], Guillermo Leguizamón [a]

[a] LIDIC, Departamento de Informática, Universidad Nacional de San Luis, Ejército de los Andes 950, 5700 San Luis, Argentina
[b] Lenguajes y Ciencias de la Computación, University of Málaga, Campus de Teatinos s/n, 29071 Málaga, Spain
[c] Faculty of Electrical Engineering and Computer Science, VŠB-Technical University of Ostrava, Czech Republic

ARTICLE INFO

Keywords:
Distributed Differential Evolution
Non-parametric tests
Benchmarking

ABSTRACT

This paper presents a new distributed Differential Evolution (dDE) algorithm and provides an exhaustive evaluation of it by using two standard benchmarks. One of them was proposed in the special session of Real-Parameter Optimization of CEC'05, and the other was proposed in the special session of Large Scale Global Optimization of CEC'08. We statistically validate and compare our results versus all other techniques presented in these special sessions. This means that more than 25 problems, with different dimensions: 30, 50, 100, and 500 variables, are evaluated; and 15 algorithms are compared in the experiments. Our dDE is simple, accurate, and competitive when applied to a wide variety of problems, with scaling dimensions, and different function features: noisy, non-separable, multimodal, rotated, etc.

1. Introduction

Metaheuristics [1,2] are optimization algorithms that allow experts to tackle complex problems by iteratively trying to improve a candidate solution, with regard to a given measure of quality (fitness). Possibly, the main feature of metaheuristics is that they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, although metaheuristics do not guarantee an optimal solution is ever found, they are usually able to obtain high quality solutions with moderate computational cost.

In the last decade, Differential Evolution (DE) [3,4] has emerged as a prominent metaheuristic for multidimensional real-valued functions. This technique was designed by Storn and Price in 1997 and has attracted a great attention from the research community since, it is simple and easy to understand, and it shows a special ability to deal with non-differentiable and multimodal optimization problems. In addition, there exists a number of works on theoretical and practical aspects of DE (multiobjective, constrained, dynamic, parallel, etc.) and has been applied on a wide range of real-world problems [4].

Besides the temporal complexity of some (NP-Hard) problems, they can be handled by canonical sequential metaheuristics (including DE), although the exploration procedure (of the search space) performed by them is also time-consuming. In addition, the size of the problem solution space, as well as its complexity, turn increasingly larger with the number of decision variables. As a result, more efficient search strategies are required to explore all the promising regions with limited computational resources.

* Corresponding author.
E-mail addresses: javierma@unsl.edu.ar (J. Apolloni), jnieto@lcc.uma.es (J. García-Nieto).

http://dx.doi.org/10.1016/j.amc.2014.03.083
0096-3003/© 2014 Elsevier Inc. All rights reserved.

Parallel distributed models are very useful tools to improve the performance of such techniques during the search process. In particular, DE methods can be easily distributed in a parallel model, since they are based in the evolution of a population of individuals. This population can be partitioned into small subsets known as islands, each subset evolving independently from each other, usually exploring different regions. These islands are spatially structured and exchange information among them to hopefully increase the accuracy and efficiency of the resulting algorithm. When run in a parallel computer, the time reduction is an additional advantage [5]. Furthermore, the latest advances in computing architectures and telecommunication networks allow us to link computers to create a powerful tool for low-cost computing. According to this, the parallel DE algorithm we are proposing here could be used at any granularity, i.e., at the computer, core, or thread levels. For instance, each core could process an island whose set of individuals is evolved by a DE algorithm.

In this work, our main motivation is to develop and evaluate a set of distributed versions of Differential Evolution (dDE) with the aim of empirically assess whether these kinds of optimizers are competitive with the current state of the art, or not. Our conjecture is that, a periodical migration of individuals in a given topology leads to a high exploration ability in DE, since the foreign solutions could provide diversity to the island population. We are also interested to analyze the behavior of dDE versions on a set of many problems with different properties: separable/non-separable, unimodal/multimodal, shifted, rotated, hybrid/composed, and on a scaling benchmark with large dimension landscapes.

Therefore, in this work we perform a thorough experimentation with our distributed versions of Differential Evolution (dDE) by following the standard procedures applied in two well-known special sessions: Real-Parameter Optimization of CEC'05 [6] and Large Scale Global Optimization of CEC'08 [7]. We statistically assess and compare our results against all the other techniques presented in these sessions, summing up 15 efficient algorithms in the top of the state of the art. The resulted distributions lead us to claim the competitive performance of our dDE, even on specially hard problems with intricate shapes and deceptive landscapes.

The contributions of this work can be enumerated as follows:

1. We have analyzed the performance of several distributed versions of DE, and considered also a sequential canonical one. A version with two populations seems to show the best results.
2. Our proposed dDE has been evaluated in the CEC'05 experimental framework and compared with the eight proposed algorithms in this benchmark, for problem dimensions of 30 and 50 variables. These comparisons show the competitive performance of our distributed approach, statistically better than other DE versions, and similar to G-CMA-ES (the winner in this special session).
3. In an extensive experimentation, we have evaluated our dDE in the scope of CEC'08 test suite. In this benchmark, a number of 7 large scale problem functions with dimensions 100 and 500 variables are optimized. After statistical comparisons with regards to all presented algorithms in this special session, we can observe that our dDE is again located in the top level of techniques with the best performances.
4. Further analysis concerning the problem function properties show the ability of our dDE to obtain an excellent behavior on non-separable and multimodal functions.

The remaining of this paper is organized as follows. Next section offers a review of distributed DE approaches found in the current literature. In Sections 3 and 4, the Canonical DE and the parallel distributed model proposed in this study are described, respectively. Experiments, comparisons, and analysis are presented in Section 5. Finally, concluding remarks and future work are provided in Section 6.

2. Literature overview: distributed DE approaches

This section presents a brief overview of the main existing works dealing with distributed population DE algorithms in the literature. Practically all of them consist on island distributed models, with different topologies, and trying to induce diversification/intensification search mechanisms by means of the migration policy of solutions. It is worth mentioning that the use of parallel resources to improve the computational cost is an additional advantage only exploded in some of these works.

A first approach was proposed by Zaharie and Petcu [8] consisting on a distributed DE, in which the population is divided into several sub-populations and one DE algorithm is executed in parallel on each sub-population. The motivation of this work was to tackle each optima in multimodal problems with different DE islands, following a random topology for solution exchange. A number of 6 functions were solved by this method. After this, Tasoulis et al. [9] proposed a dDE with islands physically assigned to different processors. In this proposal, a ring topology is established to connect islands and different mutation strategies were also analyzed in the scope of 7 optimization functions and few dimensions (2 to 30). Following this parallel scheme Kozlov and Samsonov [10] used a dDE approach to optimize segment determination gene network, although making a slight adaptation to the migration policy. A modified version of distributed DE was developed by De Falco et al. [11,12] for the registration of 2-D satellite images and for determining the optimal mapping of resource on grid computing, respectively. This proposal used slave processors to execute DE instances and one master processor to collect the information concerning each island. After this, Apolloni et al. [13] made a first preliminary analysis on a bi-population DE with encouraging results that were however partial and need a clear extension on the benchmark used and the algorithms studied to confirm their initial findings.

In the above cases, the islands execute identical instances of algorithms with identical parameter setting, then it is said that is an homogeneous distributed model. On the contrary, the following proposals use different parameter setting for each DE island, so the distributed model is featured as heterogeneous.

A generic parallelization framework, with several DE mutation strategies, for global optimization was introduced by Izzo et al. [14]. In this approach, the distribution model consisted on up to 5 islands connected by an asynchronous ring topology. The experiments in the scope of 5 well-known problem functions revealed the high performance of the proposal with regards to sequential DE. A novel heterogeneous dDE strategy was proposed by Weber et al. [15], where the population is also partitioned to constitute bi-population models. The sub-populations of the two groups evolve using an island-based DE algorithm, but one of them is used as external archive to keep the population diversity. In particular, at the end of the process, the best solutions are exchanged to incorporate diversity in these foreign sub-populations.

In [16] an heterogeneous ring topology island DE model was introduced. In this algorithm, each island was configured with its own mutation factor parameter that may be perturbed as consequence of the migration process. The authors argued that this adaption method can be used for improving the exploration and exploitation of the search space, and tested it on a well-grounded experimental framework. Tasgetiren et al. [17] proposed an ensemble of distributed DE where each island has its own crossover operator and its own parameter values. After each evolutionary step, all of the sub-populations are merged in a single population from which are selected the solutions of the next step. In [18], a different way to perturb the mutation parameter and to improve the exploration and exploitation of the search space was proposed. More specifically, this proposal employs a mechanism that collects all of the sub-populations into a single population and then, divides randomly the population into new sub-populations.

More recently, in [19] a thorough experimentation was carried out to evaluate and compare a number of different DE versions. In this study, distributed DE versions resulted with a better performance than panmictic ones, as well as other DE approaches in the state of the art. The influence of specific mutation operators were also investigated in this work. A pool of strategies was used by Bujok and Tvrdík [20] to develop a competitive island-based DE. In this approach, the islands are connected into a star topology, each of them executing an instance of DE algorithm. A series of new solutions were generated from operators selected from the pool of strategies. A parallel self-adaptive proposal was introduced by Xie et al. [21], where islands are connected through a ring topology. In this last work, each island uses a DE algorithm to self-adapt the parameter values and avoid getting trapped into local optimal solution.

From a different point of view, some authors introduced proposals that modify the topological connection during the evolutionary process. The goal is again to preserve population diversity and avoid premature convergence. In this regard, Biazzini and Montresor [22] proposed a dynamic topology of islands, where DE algorithms are randomly linked anew after each migration operation. The agents attempt to establish which sub-population contributes the most to the evolution, and employ this information to dynamically adapt the connection between islands. A recent strategy in this sense was presented by Sun et al. [23]. In this proposal, a parallel dDE is enhanced with an ensemble of different topologies that automatically adapt the connections among islands during the evolutionary process.

The above studies propose several interesting approaches focused on different design aspects of the DE distribution/parallelization. In the present work, an extensive experimentation is performed with the aim of shedding light on how competitive our dDE is, with regards to other outstanding DE versions, as well as other different metaheuristics in the state of the art. In concrete, the following features can be mentioned that make our work different to other distributed DE studies in the literature:

1. They tackled limited sets of functions with few scaling dimensions. Practically all of them used around 6 well-known functions (Ackley, Rastrigin, Rosenbrock, etc.) with up to 50 problem variables. We use here an extensive benchmark (more than 25 functions) with scaling number of variables: 30, 50, 100, and 500.
2. They where compared against other new and existing versions of DE. In our work, experimental comparisons are addressed in the scope of prominent competitors in special sessions (CEC'05 and CEC'08), where sophisticated versions of DE and other metaheuristics were also tested.
3. They analyzed the performance induced by adapting parameters and/or operators, mostly focused on migration strategies. In the present work, a specific analysis concerning the different function properties is carried out. Then, the possible advantage of using distributed DE populations on: non-separable, multimodal, shifted, and rotated functions is investigated.

3. Differential Evolution: background

In the next section, our island model dDE will be described. Before, in this section, background concepts of canonical Differential Evolution algorithm are explained.

Differential Evolution (DE) [4] is a stochastic population-based algorithm initially designed to solve optimization problems in continuous domains. In DE, the population is a set of individuals (tentative solutions) which evolve simultaneously through the search space of a given problem to optimize. The individuals are real-value vectors that, by combinations with others from the population, generate new individuals.

DE tackles an optimization problem by maintaining a population of candidate solutions and updating them according to a scoring function or fitness. This function provides each solution with a quality measure relative to the optimization problem at hand. In this way, the optimization problem is treated as a black box and additional gradient information is not needed, as happens with classic optimization techniques such as gradient descent and quasi-newton methods [24].

Formally, the population $P = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is a set of N individuals represented by means of real-valued vectors:

$$\mathbf{x}_i^t = (x_i^t(1), x_i^t(2), \ldots, x_i^t(D)),$$

where $x_i^t(j) \in [x_{low}, x_{upp}]$ with x_{low}, $x_{upp} \in \mathbb{R}$ the lower and upper bound of the domain of the variable j $(1 \leqslant j \leqslant D)$. The current generation is represented by t $(1 \leqslant t \leqslant t_{max})$, and t_{max} is the maximum number of generations in the iterative evolutionary process.

In DE, the task of generating new individuals (real-valued vectors) is performed by operators such as the differential mutation (also known as "perturbation") and crossover. A *mutant individual* $\mathbf{w}_i^{t+1}$ is generated by a perturbation scheme that is selected when the algorithm is constructed. Storn and Price [3] originally proposed four perturbation schemes:

- *DE/rand/1*:

$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{x}_{r1}^t + F \cdot \left(\mathbf{x}_{r2}^t - \mathbf{x}_{r3}^t\right). \tag{1}$$

- *DE/best/1*:

$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{b}^t + F \cdot \left(\mathbf{x}_{r1}^t - \mathbf{x}_{r2}^t\right). \tag{2}$$

- *DE/best/2*:

$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{b}^t + F \cdot \left(\mathbf{x}_{r1}^t + \mathbf{x}_{r2}^t - \mathbf{x}_{r3}^t - \mathbf{x}_{r4}^t\right). \tag{3}$$

- *DE/rand-to-best/1*:

$$\mathbf{w}_i^{t+1} \leftarrow \mathbf{x}_i^t + \lambda \cdot \left(\mathbf{b}^t - \mathbf{x}_i^t\right) + F \cdot \left(\mathbf{x}_{r1}^t - \mathbf{x}_{r2}^t\right), \tag{4}$$

where $r1, r2, r3, r4 \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$ are random integers mutually different, and also different from index i. Individual $\mathbf{b}^t$ represents the best solution found so far. The mutation constant $F > 0$ stands for the amplification of the difference between the individuals $\mathbf{x}_{r's}$ and it avoids the stagnation of the search process. Parameter λ controls the greediness of the fourth scheme. To reduce the number of control variables, these two last parameters are usually set to the same value $(F = \lambda)$.

Algorithm 1. Pseudocode of Canonical DE

```
 1: initialize($P$)
 2: while $t < t_{max}$ do
 3:   for each individual $i$ of $P$ do
 4:     choose mutually different $r_s$ values
 5:     $\mathbf{w}_i^{t+1} \leftarrow$ mutation($\mathbf{x}_{r_s}^t, F$) // Eq. (1) or Eq. (2) or Eq. (3) or Eq. (4)
 6:     $\mathbf{u}_i^{t+1} \leftarrow$ crossover($\mathbf{x}_i^t, \mathbf{w}_i^{t+1}, Cr$) // Eq. (5)
 7:     evaluate($\mathbf{u}_i^{t+1}$)
 8:     $\mathbf{x}_i^{t+1} \leftarrow$ selection($\mathbf{x}_i^t, \mathbf{u}_i^{t+1}$)// Eq. (6)
 9:   end for
10: end while
```

In order to increase even more the diversity in the population, each mutated individual undergoes a crossover operation with the *target individual* $\mathbf{x}_i^t$, by means of which a *trial individual* $\mathbf{u}_i^{t+1}$ is generated. A randomly chosen position is taken from the mutant individual to prevent that the trial individual replicates the target one.

$$u_i^{t+1}(j) \leftarrow \begin{cases} w_i^{t+1}(j) & \text{if } r(j) \leqslant Cr \text{ or } j = j_r, \\ x_i^t(j) & \text{otherwise.} \end{cases} \tag{5}$$

As shown in Eq. (5), the crossover operator randomly chooses a uniformly distributed integer value j_r and a random real number $r \in [0, 1]$, also uniformly distributed for each component j of the trial individual $\mathbf{u}_i^{t+1}$. Then, the crossover probability Cr, and r are compared just like j and j_r. If r is less or equal to Cr (or j is equal to j_r) then the jth element of the mutant individual is selected to be allocated in the jth element of the trial individual $\mathbf{u}_i^{t+1}$. Otherwise, the jth element of the target individual $\mathbf{x}_i^t$ becomes the jth element of the trial individual.

Finally, a selection operator decides on the acceptance of the trial individual for the next generation if and only if it yields a reduction (assuming minimization) in the value of the fitness function $f()$, as shown by the following Eq. (6):

$$\mathbf{x}_i^{t+1} \leftarrow \begin{cases} \mathbf{u}_i^{t+1} & \text{if } f(\mathbf{u}_i^{t+1}) \leqslant f(\mathbf{x}_i^t), \\ \mathbf{x}_i^t & \text{otherwise.} \end{cases} \tag{6}$$

Algorithm 1 shows the pseudocode of DE. After initializing the population, the individuals evolve during a number of iterations (t_{max}). Each individual is then mutated (Line 5) and recombined (Line 6). The new individual is selected (or not) following the operation of Eq. (6) (Lines 7 and 8).

Although in the literature, up to the authors' knowledge, there is no precise indication of which scheme is the best (it seems to depend to the tackled problem), nor there is a definitive indication of which values to use for parameters, Price et al. [4] proposed a series of empirically tested values. For instance, crossover probability $Cr \in [0, 1]$ must be considerably lower than 1, e.g., 0.3, in spite of that, if no convergence can be achieved, a value in $[0, 0.8]$ should be used. For many applications, a population size of $P = 10 \times D$ is a good choice, being D the problem dimension. However, we have to note that this relation for P is usually set when no restrictions concerning the maximum number of evaluations exist. Unfortunately, in this paper we deal with functions that belong at benchmarking competitions in which a bounded number of function evaluations are allowed. As a consequence, the algorithms dealing with these functions commonly set P to an value in the interval 10 to 200 [25–39], in order to avoid consuming all of the allowed function evaluations in an early evolution phase of the algorithm. For this reason, we set P to 20 individuals for all of the experiments/problems/dimensions, thus we achieve that our proposals perform a large number of evolution steps. Regarding mutation constant F, its value is usually chosen in [0.5,1], in such a way that, the higher the population size, the lower the weighting factor F.

Clearly, in spite of having DE a few parameters to tune, the success of this algorithm is highly dependent to the complex interaction of them, especially F and Cr. In this work, a systematic tuning of this two parameters has been carried out for each optimization problem. Then, a series of F and Cr combinations will be presented in Section 5.2 for the sake of a well-grounded parameter setting.

4. Parallel Differential Evolution

As commented before, like almost all the evolutionary algorithms, Differential Evolution suffers from some drawbacks such as risking of stagnation of searching process, because it does not progress for finding any better solution, and related to this, premature convergence to sub-optimal solutions due to a quick lost population diversity.

A way of alleviating these problems is to find a balance in the explorative and exploitative capabilities of the algorithm [19]. In consequence, some components of DE must be modified for helping to keep the balance.

It is known that modifying the original structure of the population leads to improve the performance of the evolutionary algorithms, as well as other population-based metaheuristics [5]. In the literature, there exist several ways of structuring the population, e.g., cellular and island models, that, in certain complex problems, have proven to find better solutions [5,40].

Specifically, in the island model, the population is divided into small disjoint sub-populations and each sub-population is assigned to an island. This model defines a distributed algorithm since, the islands are independent and executes its own evolutionary algorithm. Besides, the island model uses a mechanism of exchanging of information during the evolutionary process. A topological structure of connecting islands, e.g., a ring, a star or randomly (see Fig. 1), is employed to send/receive solutions to/from other islands with certain frequency. It is worth noting this migration process becomes in an effective mechanism to keep the population diversity.

Our work is focused on the optimization of continuous complex functions by using a parallel distributed DE based on an implementation suggested by Tasoulis et al. [9]. We use an island connection model where the population is partitioned in small groups of individuals. The individuals inside each island evolve independently from the rest of islands, although making each island occasional communication operations with the others islands to interchange solutions.

The interchange of solutions is determined by the *migration rate*, that defines the number of individuals that are sent to (received from) other islands. A topology neighborhood is defined in the *migration policy* in order to carry out a guided exchange of solutions between subpopulations. For this migration of solutions, we fix it every certain number of steps of the evolution process of each island. For updating the islands, synchronous and asynchronous modes can be performed. Our dDE

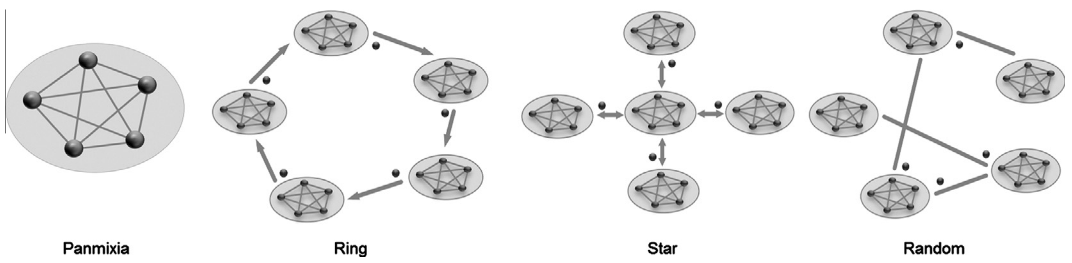

Fig. 1. Island based distribution topologies in population EAs.

has been configured as asynchronous updating since individuals are received whenever they arrive, with no stops in the execution.

4.1. Island based model of dDE

In Algorithm 2, the pseudocode of our island model distributed DE is shown. In this algorithm, the whole population P is structured in m smaller subpopulations P_i of n_i individuals where $N = \sum_{i=1}^{m} n_i$. Each subpopulation is randomly (uniform) initialized, and it evolves in parallel independently and relatively isolated from the others performing periodical exchanges of solutions.

The migration policy is determined by a five-tuple $\mathcal{M} = \langle \gamma, \rho, \phi_s, \phi_r, \tau \rangle$ where $\gamma \in \mathbb{N}$ denotes the migration gap between two successive exchanges of individuals, $\rho \in \mathbb{N}$ denotes the migration rate in every exchange, functions ϕ_s and ϕ_r decide how to select the individuals involved in the exchange. The *selection function* ϕ_s, decides what individuals select to migrate, and the individuals that arrive substitute those local individuals previously determined by the *replacement function* ϕ_r. The topological model is denoted by the function $\tau : P \to 2^P$, which selects what subpopulations can send to (or receive from) individuals.

Algorithm 2. Pseudocode of the Distributed DE

```
1: DO IN PARALLEL for each $i \in \{1, \ldots, m\}$
2: initialize($P_i$)
3: while $t < t_{max}$ do
4:    perform a step of canonical DE// as in Algorithm 1
5:    for each of the $\rho$ individuals to send do
6:       $\mathbf{v}^t \leftarrow \phi_s(P_i)$
7:       send $\mathbf{v}^t$ to $P_j$ chosen by $\tau$ // being $P_j$ the neighbor subpopulation
8:    end for
9:    while individuals are arriving do
10:      receive $\mathbf{v}^t$ /*asynchronous communication*/
11:      replace individual chosen from $\phi_r(P_j)$ by $\mathbf{v}^t$
12:   end while
13: end while
```

In our algorithm, the individuals to be migrated are uniform randomly chosen by the selection function ϕ_s (line 6 in Algorithm 2). If the incoming individuals from other islands have higher performance than local ones chosen by the replacement function ϕ_r (line 11 in Algorithm 2), then the latter are substituted by the former. The topology is a unidirectional *ring* in which the individuals are exchanged with the nearest neighbor subpopulation.

5. Experimental study

We now analyze the behavior of our dDE by performing a set of experiments plus a statistical validation with the reported results. We have used the skeleton architecture in C++ of the MALLBA Library [41] to easily develop our dDE algorithm. The underlying communication platform is implemented with the MPICH library (v.1.5.2) deployed on machines with O.S. Linux SUSE.

For the experimental comparisons, we have used non-parametric statistical tests, since some times the numerical distributions of results did not follow the conditions of normality and homoscedasticity [42]. Therefore, our analyses are mainly focused on the whole distribution errors, although paying special attention on the mean errors, out of 25 independent runs. In particular, we have considered the application of Wilcoxon' Pairwise, Friedman's ranking, and Holm's multicompare tests, this last as post hoc procedure [43], to know which algorithms are statistically worse than the reference algorithm (the one with the best ranking).

We include in this study a series of comparisons with the canonical version of DE, as well as the algorithms of the state of the art (reference techniques for the tackled problems) in order to clarify how competitive our proposal is. Before, the set of benchmarking test functions and the parameter settings are detailed.

5.1. Test functions

As commented in Section 1, we have used two different sets of functions to tests our dDE algorithm: the benchmark proposed in special session on Real-Parameter Optimization of CEC'05 [6], and the benchmark proposed in the special session on Large Scale Global Optimization of CEC'08 [7]. The test suite CEC'05 includes 25 optimization functions with different properties. The first 5 functions are unimodal and the remaining (20) are multimodal ones. In this work, we are mainly interested

Table 1
CEC'05 and CEC'08 benchmarks with functions' features: unimodal (U), multimodal (M), separable (Sep.) and non-separable, rotated (Rot.) and non-rotated. Problem search ranges (S.R.) and biases to optima values f^* are specified. RHC stands for Rotated Hybrid Composed.

f	Name	U/M	Sep.	Rot.	S.R.	f^*
f_6	Shif. Rosenbrock's Function	M	N	N	[−100, 100]	−390
f_7	Shif. Rot. Griewank's. Opt. Out. Bounds	M	N	R	[0, 600]	−180
f_8	Shif. Rot. Ackley's Opt. on Bounds	M	N	R	[−32, 32]	−140
f_9	Shif. Rastrigin's Function	M	S	N	[−5, 5]	−330
f_{10}	Shifted Rotated Rastrigin's	M	N	R	[−5, 5]	−330
f_{11}	Shifted Rotated Weierstrass	M	N	R	[−0.5, 0.5]	90
f_{12}	Schwefel's Problem 2.13	M	N	N	$[-\pi, \pi]$	−460
f_{13}	Shif. Exp. Griewank's $\bigoplus$ Rosenbrock's	M	N	N	[−3, 1]	−130
f_{14}	Shif. Rot. Expanded Scaffer's F6	M	N	R	[−100, 100]	−300
f_{15}	HC (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	N	[−5, 5]	120
f_{16}	RHC f15	M	N	R	[−5, 5]	120
f_{17}	F16 with Noise in Fitness	M	N	R	[−5, 5]	120
f_{18}	RHC (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	R	[−5, 5]	10
f_{19}	RHC Narrow Basin Global Optimum	M	N	R	[−5, 5]	10
f_{20}	RHC Global Optimum on Bounds	M	N	R	[−5, 5]	10
f_{21}	RHC (f1-f2,f3-f4,f5-f6,f7-f8,f9-f10)	M	N	R	[−5, 5]	360
f_{22}	RHC High Condition Number Matrix	M	N	R	[−5, 5]	360
f_{23}	Non-Continuous Rot. Hybr. Comp.	M	N	R	[−5, 5]	360
f_{24}	RHC (f1,f2,f3,f4,f5,f6,f7,f8,f9,f10)	M	N	R	[−5, 5]	260
f_{25}	RHC Global Optimum Outside Bounds	M	N	R	[2, 5]	260
g_1	Shifted Sphere Function	U	S	N	[−100, 100]	−450
g_2	Shifted Schwefel's Problem 2.21	U	N	N	[−100, 100]	−450
g_3	Shifted Rosenbrock's Function	M	N	N	[−100, 100]	390
g_4	Shifted Rastrigin's Function	M	S	N	[−5, 5]	−330
g_5	Shifted Griewank's Function	M	N	N	[−600, 600]	−180
g_6	Shifted Ackley's Function	M	S	N	[−32, 32]	−140
g_7	FastFractal "DoubleDip" Function	M	N	N	[−1, 1]	Unknown

on multimodal functions with high complexities. The CEC'08 test suite includes 2 unimodal and 5 multimodals. The first 6 are basic and composed functions and the seventh one has been taken from an external benchmark of fractal functions whose global optimum is unknown so far.

Table 1 shows the set of functions used in this study. A total number of 27 functions with heterogeneous landscapes and complexities, which are indexed as: f_6 to f_{25} corresponding to CEC'05 functions, and g_1 to g_7 the ones of CEC'08. In terms of function structure, they can be featured as: single shifted structure (f_6 to f_{12}), expanded (f_{13} to f_{14}), hybrid composed (f_{15} to f_{25}), and large scale (g_1 to g_7). In this table, their most interesting properties are also highlighted: unimodal (U), multimodal (M), separable (Sep.), non-separable, shifted to biased optimum, rotated (Rot.), and hybrid composed. The respective bounds of search ranges (S.R.) and biases to optima (f^*) are also indicated. In this regard, all functions except for g_7 (which optimum is unknown) have the global optimum shifted to a value different from zero named *bias*. A shifted function to a bias is useful to avoid a symmetry on the search space that DE could exploit in its benefit. In two functions (f_8 and f_{20}), the optima cannot be found within the initialization range, and the domain of search is not limited (the optimum is out of the range of initialization). The complete descriptions of all these functions can be found in [7,6].

With the aim of illustrating the hardness of the tackled problems, Fig. 2 shows the fitness landscapes of functions: f_{15}, f_{19}, g_6, and g_7. All these functions, excepting g_6, show intricate landscapes with a great number of basins of attraction to local optima, with multi-funnel regions, and with rotated structures to the axis of coordinates. In the case of g_6, the landscape is equally complex, since it consists on a great plateau area (where is difficult to search) with a narrow and shifted basin of attraction near to the borders. We have to note that these fitness landscapes were plotted for just two problem variables (dimension 2), although we have dealt with: 30, 50, 100, and 500 variables in our experiments.

Following the specifications of the two benchmarks used, we have applied as stop conditions a maximum number of $10,000 \times D$ fitness evaluations for CEC'05 functions, with scaling dimensions $D = 30$ and $D = 50$; and $5000 \times D$ fitness evaluations for CEC'08 functions, with $D = 100$ and $D = 500$. We have performed 25 independent runs for each investigated DE version, problem function, and problem scale, summing up $25 \times 3 \times 27 \times 4 = 8100$ experimental executions. We report the error values of the best solution (x) found defined as: $f(x) - f^*$, where f^* is the optimum fitness of function f. Error values lower than 10^{-14} (*0-threshold*) are approximated to zero.

5.2. Parameter setting

The set of parameters used to tune our dDE were selected after preliminary experiments as follows: the population P was set to 20 individuals and was partitioned in m subpopulations (islands), each one having n individuals where $\sum_m n = 20$. The

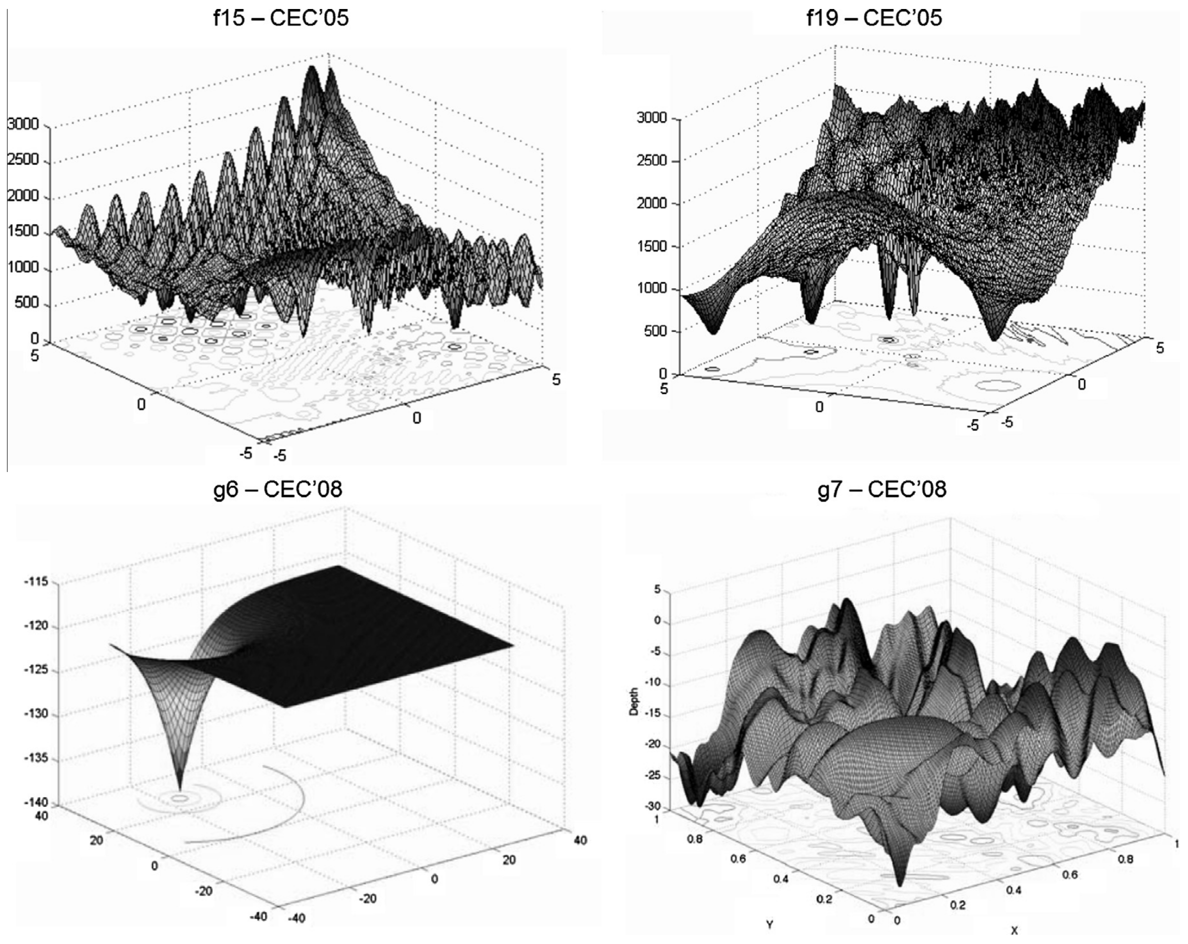

Fig. 2. Complex fitness landscapes of functions: f_{15},f_{19},g_6, and g_7.

Table 2
Parameters *F* and *Cr* of dDE for each problem function of CEC'05 benchmark.

F/D	30		50	
	F	*Cr*	*F*	*Cr*
f_6	5.0E−01	4.0E−01	5.0E−01	4.5E−01
f_7	5.0E−01	5.0E−01	5.0E−01	5.0E−01
f_8	5.0E−01	5.0E−01	5.0E−01	5.0E−01
f_9	9.0E−01	1.0E−02	9.0E−01	1.0E−02
f_{10}	5.0E−01	2.0E−01	4.0E−01	5.0E−01
f_{11}	9.0E−01	5.0E−01	9.0E−01	1.0E−01
f_{12}	5.0E−01	1.0E−01	5.0E−01	1.0E−01
f_{13}	9.0E−01	1.0E−03	9.0E−01	1.0E−03
f_{14}	9.0E−01	1.0E−03	9.0E−01	1.0E−03
f_{15}	1.0E−01	1.0E−03	9.0E−01	1.0E−03
f_{16}	9.0E−01	1.0E−02	9.0E−01	2.0E−01
f_{17}	9.0E−01	5.0E−01	9.0E−01	5.0E−01
f_{18}	5.5E−01	4.0E−01	5.5E−01	4.0E−01
f_{19}	5.0E−01	5.0E−01	5.0E−01	5.0E−01
f_{20}	5.5E−01	5.0E−01	5.5E−01	5.0E−01
f_{21}	5.0E−01	1.0E−02	5.0E−01	1.0E−01
f_{22}	5.0E−01	5.0E−01	5.0E−01	5.0E−01
f_{23}	5.0E−01	1.0E−01	5.0E−01	1.0E−02
f_{24}	9.0E−01	9.0E−01	9.0E−01	9.0E−01
f_{25}	9.0E−01	9.0E−01	9.0E−01	9.0E−01

migration gap and migration rate were set to $\gamma = 100$ and $\rho = 1$, respectively. As previously mentioned, one randomly selected individual is sent from an island to another one in a non-blocking policy of migration.

Concerning the DE specific parameters, Table 2 shows the values of F and Cr. These values have arisen from a systematic adjust that we performed for each function and dimension of the benchmark suite CEC'05. In the case of CEC'08, parameters have been set to F = 5.0E−01 and Cr = 1.0E−02 for all functions and dimensions, in order to following the competition rules of this benchmark [7]. It is worth noting that all of these values have been fixed for all our DE versions, including the canonical one. In this way, we look for applying the same learning procedure to our Differential Evolution strategy, with the aim of discovering the underlying benefits of using different distribution models.

5.3. Comparative study

This section presents a comparative study of our different dDE versions. Later, a series of analyses in terms of function's properties will be also given.

5.3.1. Comparative study of dDE versus Canonical DE

As a first analysis, we compare here the canonical version of DE (seqDE) with two different versions of our island model DE (dDE). In the former, a whole population of 20 individuals evolves without separated structures. In the first island model, dDE_2, the population of 20 individuals is split into two subpopulations (islands) of 10 individuals each one. In the second island model, dDE_4, the whole population is divided into four subpopulations of 5 individuals.

For these comparisons, we have applied the Wilcoxon Signed-Rank test [44] to compare each possible pair of DE models (seqDE, dDE_2, dDE_4) on CEC'05 functions and for dimensions with 30 and 50 problem variables. As shown in Table 3, the statistical test calculates the differences between two distributions, and these differences are split into groups of positive and negative values. In this table, column R+ represents the sum of ranks with positive differences, and R− is the respective sum of negative differences, which gives us information about the algorithm with better performance. For example, in the comparison of seqDE versus dDE_2 for dimension 30, the sum of positive ranks (105) is greater than the sum of negative ones (48), meaning that seqDE distributions are higher than the ones of dDE_2. Since we are dealing with minimizing functions, we can claim that our distributed DE with 2 islands is better ranked than the Canonical DE (seqDE), for CEC'05.

With the aim of showing whether statistical differences exist (or not) between these distributions, we have also calculated the p-values of these tests with a confidence level of 95%. In this way, we can assure that significant differences exist when resulted p-values are lower than 0.05.

As we can observe in the Table 3, our distributed DE model with two island (dDE_2) achieves higher ranks than the canonical model (seqDE) for the two dimensions: 30 and 50. A similar observation can be made when comparing the Canonical DE and the distributed model with four islands (dDE_4) for dimension 50. Although in the case of dimension 30, the sum of the positive ranks of seqDE is lower than the sum negative ones, then seqDE outperforms the performance of dDE_4. However, the most interesting results are the ones concerning the comparison of the two distributed models, since for all dimensions, dDE_2 shows better ranks than dDE_4, and resulted p-values are in both cases lower than 0.05 (2.00E−03 for dimension 30 and 1.80E−02 for dimension 50).

From a graphical point of view, Fig. 3 shows the trace progress performed by the three DE versions: seqDE, dDE_2, and dDE_4, for functions f_6,f_{12},f_{17}, and f_{22} with dimension 30 variables. In this plot, we can observe that the distributed versions of DE show, in general, a delayed convergence with regards to the canonical one, probably induced by the migration policy in the formers. We suspect that the migration of fresh individuals from the other island in dDE_2 could enhance the local diversity of subpopulations, then improving the global results in terms of solution quality. However, an excessive diversity together with a low number of individuals in subpopulations could degrade the final results, as observed in dDE_4.

We now will face the last analysis of this section, where a comparison involves the dDE approach proposed in [9] since, as we commented before, we have based on this algorithm to design our own distributed Differential Evolution.

For the reproduction of this approach (Tasoulis et al. [9]), we have followed the specifications as described in the referenced work, where the authors used a similar island-distribution scheme to our dDE_2. For the sake of a fair comparison, we use two population and set P to 20 individuals, i.e., 10 individual in each population, and the parameter values for F and Cr are taken from Table 2. However, in Tasoulis' proposal we use the differential mutation *DE/best/1* (Eq. (2)) because it was

Table 3
Signed-Rank test of seqDE, dDE_2 and dDE_4 in terms of the mean of the error value at a significance level of 95% (p-value = 0.05). Lowest signed-ranks are shown in bold.

Algorithm	Dimension	R+	R−	p-value
seqDE versus dDE_2	30	105	**48**	1.85E−01
	50	110	**61**	2.95E−01
seqDE versus dDE_4	30	**66**	105	4.08E−01
	50	107	**103**	9.56E−01
dDE_2 versus dDE_4	30	**15**	156	2.00E−03
	50	**31**	140	1.80E−02

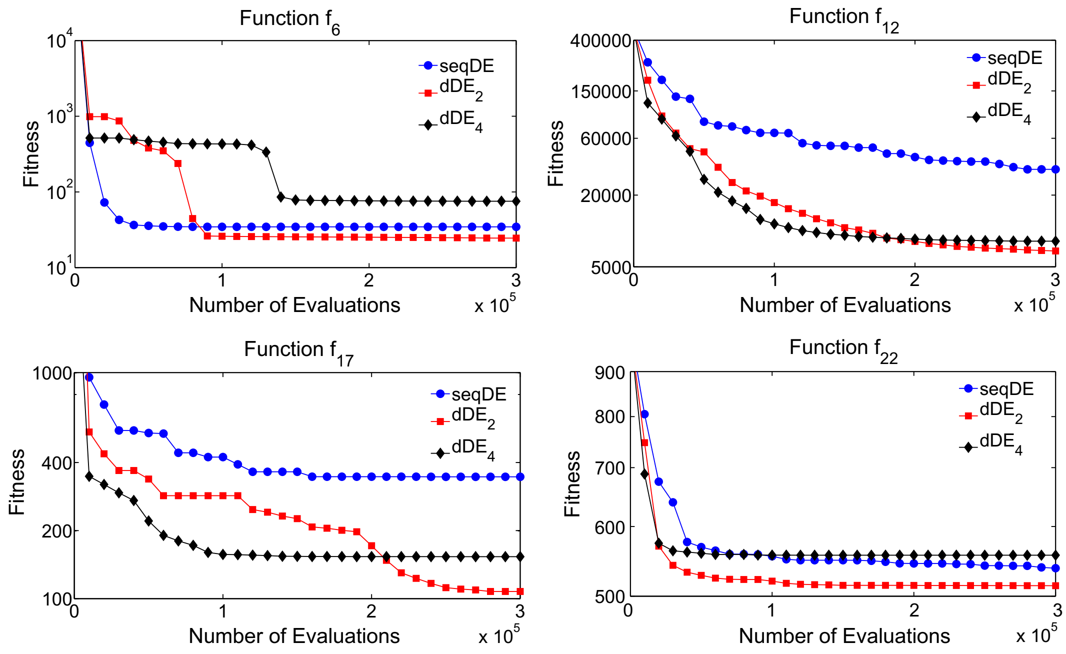

Fig. 3. Function evaluations versus error value ($(f(x) - f(x^*))$), where x is the best solution found so far and x^* is the best solution known (performance of the median of 25 runs).

shown to be the best option in [9]. In addition, differently to our proposal, in [9] the migration of best individuals is controlled by the migration constant, $\phi \in [0,1]$. According to this, at each iteration, a uniformly distributed random number in the interval $[0,1]$ is chosen and compared with the migration constant. If the migration constant is larger, then the best individuals of each subpopulation migrate and respectively take the place of a randomly selected individual (different from the best) in the next subpopulation; otherwise no migration is done.

Table 4 shows the results of applying the signed-rank (Wilcoxon) statistical test to the resulted distributions of our dDE$_2$, in comparison to the ones of [9], in the scope of CEC'05 functions and for dimensions 30 and 50. In this table, it is clearly observable that dDE$_2$ is better ranked than Tasouli's dDE with statistical differences for 30, as well as for 50 problem variables. The resulted p-value is 0.00 for both dimensions, meaning that for all functions in CEC'05, our dDE$_2$ obtained the best mean results, with regards to the compared approach. A possible reason of such differences in results could be due to the elitist behavior that *DE/best/1* mutation used in [9], which together with a high migration frequency (close to 25000 migrations for dimension 50, in contrast with 250 in the case of our dDE$_2$), would induce a premature lost of diversity in the two island subpopulations, then leading this algorithm to quickly converge to local optima in CEC'05 functions.

Therefore, the statistical results obtained by this a priori analysis are useful to suggest the selection of our dDE$_2$ as the base proposal to beat other existing algorithms in the literature.

5.3.2. Comparison with CEC'05 algorithms

In this section, once we have selected dDE$_2$ as our proposal, i.e., distributed DE with two islands, we go one step beyond to validate the behavior of this algorithm in the context of the standard protocol of CEC'05, and with regards to other competitive techniques in the state of the art. These techniques were all participant algorithms in that special session and consist of modern real-coded optimizers, following evolutionary computation or swarm intelligence paradigms, and some of them hybridized with local search or using memory-based methods.

In concrete, these optimizers are: BLXMA [25] Real-Coded Memetic Algorithm, BLX-GL50 [26] Hybrid Real-Coded Genetic Algorithm with Female and Male Differentiation, CoEVO [27] Cooperative Evolution EA, Canonical DE [28], G-CMA-ES [29] Covariance Matrix Evolution Strategy and Restarting method, K-PCX [30] Steady-State Evolutionary Algorithm, L-CMA-ES [31] Covariance Matrix Evolution Strategy Improved with Local Search, SPC-PNX [32] Steady-State Genetic Algorithm.

Table 4
Signed-Rank test of dDE$_2$ and the proposed dDE in [9], in terms of the mean of the error value at a significance level of 95% (p-value = 0.05). Lowest signed-ranks are shown in bold.

Algorithm	Dimension	R+	R−	p-value
dDE$_2$ versus Tasoulis et al. [9]	30	210	**0**	0.00
	50	210	**0**	0.00

Table 5
Mean errors obtained by dDE$_2$ for CEC'05 functions and for dimensions: 30 and 50.

Function	Mean error	Function	Mean error
Dimension 30			
f_6	2.16E+02	f_{16}	2.26E+02
f_7	3.76E−02	f_{17}	1.50E+02
f_8	2.10E+01	f_{18}	8.22E+02
f_9	1.19E−01	f_{19}	8.27E+02
f_{10}	9.04E+01	f_{20}	8.22E+02
f_{11}	3.90E+01	f_{21}	5.00E+02
f_{12}	7.27E+03	f_{22}	5.16E+02
f_{13}	9.12E−01	f_{23}	5.74E+02
f_{14}	1.28E+01	f_{24}	2.24E+02
f_{15}	1.43E+02	f_{25}	2.12E+02
Dimension 50			
f_6	1.18E+02	f_{16}	1.51E+02
f_7	3.80E−03	f_{17}	1.46E+02
f_8	2.12E+01	f_{18}	8.39E+02
f_9	7.96E−02	f_{19}	8.55E+02
f_{10}	2.05E+02	f_{20}	8.41E+02
f_{11}	5.11E+01	f_{21}	7.27E+02
f_{12}	2.96E+04	f_{22}	5.00E+02
f_{13}	1.80E+00	f_{23}	7.09E+02
f_{14}	2.26E+01	f_{24}	3.46E+02
f_{15}	1.04E+02	f_{25}	2.68E+02

Table 5 shows the mean error values obtained by dDE$_2$ (out of 25 independent runs) for all multimodal functions of CEC'05, and for dimensions 30 and 50 variables. We provide these results for the sake of the experimental reproduction and also to make them useful for future comparative studies. Then, using this distribution of results and those of compared algorithms, we have applied a statistical procedure consisting of a Friedman's ranking test and a Holm's post hoc correction (with confidence level $\alpha = 0.05$).

In Table 6, we can observe the results of applying these two statistical tests to the aforementioned distributions, for all the compared algorithms and for dimensions 30 and 50 variables. The third column in this table (*F.Rank*) contains the ranking values, sorted from best (minimum) to worst (maximum), obtained after applying the Friedman test. The best ranked algorithm by this test is then used as control sample for the Holm's post hoc correction, which results in terms of Adjusted p-values (*Holm's* A_p) are shown in the fourth column. As we can see in this table, for dimension 30, our dDE$_2$ is ranked in the top of best algorithms, and it does not show statistical difference with regards to G-CMA-ES, the control algorithm. An interesting observation is that Canonical DE is in the bottom part of this ranking (followed by CoEvo) and with a Holm's correction of 6.99E−02. This means that Canonical DE is statistically outperformed by the control algorithm since its adjusted p-value (6.99E−02) is lower than 0.05, the confidence level. In contrast, our two-islands dDE performs a similar behavior to G-CMA-ES, the best ranked algorithm in CEC'05 for dimension 30.

Nevertheless, these last results are still improved for dimension 50, in which, only two algorithms (G-CMA-ES and L-CMA-ES) were applied to these problems of CEC'05 with high complexity. For this problem scale (see Table 6 dimension 50), our dDE$_2$ obtains the best rank in comparison with the two CMA-ES versions and is given as the control algorithm,

Table 6
Average Friedman's rankings with Holm's Adjusted p-values ($\alpha = 0.05$) for CEC'05 functions, with dimensions: 30 and 50 variables. Symbol * indicates the Control algorithm. In bold, our proposal's results.

Dimension	Algorithm	*F. Rank*	*Holm's* A_p
30	*G-CMA-ES	3.67	–
	dDE$_2$	**3.89**	**1.00E+00**
	BLXMA	4.22	1.00E+00
	L-CMA-ES	4.27	1.00E+00
	BLCGL50	4.57	1.00E+00
	KPCX	4.75	1.00E+00
	SPCPNX	4.90	9.43E−01
	Canonical DE	6.52	6.99E−03
	CoEvo	8.17	1.62E−06
50	***dDE$_2$**	**1.70**	–
	G-CMA-ES	2.09	4.11E−01
	L-CMA-ES	2.22	2.27E−01

although without statistical significance in this case. We suspect that the imposition of a high significance level (95%) in the Holm's correction does not allow to show a significant difference from our dDE$_2$, but it could be easily reached for L-CMA-ES by using a significance level of 90%.

Therefore, we can state that our proposal of dDE$_2$ shows a competitive performance in high complexity problems, where only specialized versions of CMA-ES obtained successful results until now. The high specialization of CMA-ES for these problems and its complex implementation contrast with the wide applicability of our proposal and with its easy implementation and understanding.

5.3.3. Comparison with CEC'08 algorithms

In this section, we extend this study to compare our dDE$_2$ with a series of state of the art algorithms in the context of CEC'08 functions for large scale optimization. Then, we follow a similar statistical procedure to that explained before, although for dimensions 100 and 500 variables and for algorithms presented in CEC'08 special session. These techniques consist in base line metaheuristics that were adapted to perform efficiently on large scaling environments.

In concrete, these algorithms are: MTS [33] Multiple Trajectory Search algorithm, MLCC [34] Multilevel Cooperative Coevolution Algorithm, jDEdynNP-F [35] Self-Adaptive DE Algorithm, LSEDA-gl [36], Extend Univariate Estimation Distribution Algorithm, DMS-L-PSO [37], Dynamic Multi-Swarm Particle Swarm Optimization (PSO) improved with a Local Search, DEwSAcc [38] DE Algorithm Extended by Self-Adaptation of Control Parameters and a Cooperative Co-Evolution Mechanism, EPUS-PSO [39] Traditional PSO with an Efficient Utilization of the Population. We pay special attention on jDEdynNP-F and DEwSAcc, since they also use DE as base optimizer, as happens with dDE$_2$. The use of advanced mechanisms in these competitors for this kind of complex problems would contrast to the simple population distribution of our proposal.

Table 7 shows the mean error values obtained by dDE$_2$ (out of 25 independent runs) for the set of CEC'08 problem functions, and for dimensions of 100 and 500 variables. These distributions are then computed in Table 8, where the results of applying the Friedman's ranking test and the Holm's correction, with regards to all the participants in CEC'08 are shown.

A first observation in Table 8 consists in the ranking values obtained by dDE$_2$ which are in the top level for the two problem dimensions: second position for 100, as well as for 500 problem variables, and without statistical differences with regards to the control algorithm (MTS). In the case of 100 variables, dDE$_2$ results with a similar rank and *p*-value Holm's correction to DMS-L-PSO. For 500 variables, dDE$_2$ shows similar rank and *p*-value Holm's correction to jDEdynNP-F, both of them with regards to the control algorithm.

Second, in this comparison, two algorithms are statistically outperformed: MLCC and EPUS-PSO, for dimension 100. Nevertheless, for the largest problem dimension (500 variables), we cannot ensure the existence of statistical differences. In fact, we can observe that the performance of all compared algorithms become similar as the problem scale increases, so even for the top ranked algorithms (MTS and dDE$_2$), the great number of variables to be managed and their interdependencies make them to show a moderate behavior for dimension 500.

A last interesting observation in this analysis concerns other hybrid algorithms using difference-vector operations as base line optimizers (DE and PSO). In this regard, we can observe that in most of cases, dDE$_2$ shows better ranking values than jDEdynNP-F and DEwSAcc, which also use DE as main search procedure. In comparison with PSO versions, our proposal obtained similar rank to DMS-L-PSO, and outperforms EPUS-PSO with statistical confidence. In the light of this results, we can claim that our distributed population DE is also competitive on large scale complex problems, for which other hybrid algorithms with adapted operators reached lower ranking positions according to standard statistical tests.

5.4. Results in terms of function types

In this section, we present an analysis in terms of the different function features that our dDE$_2$ can successfully tackle with regards to the other compared techniques, in the scope of CEC'05 and CEC'08. Tables 9–11 show a detailed comparison presented in form of (win, tie, lose) according to different function features: Separable/non-separable, Unimodal/multimodal, Rotated/non-rotated, Shifted/non-Shifted, Scalable, and Hybridized.

A first interesting observation concerns the comparison of dDE$_2$ versus G-CMA-ES for CEC'05 functions, since they are both the best ranked algorithms for dimensions 30 and 50. In this regard, we can see in Tables 9 and 10 that our approach

Table 7
Mean results of the error values reached by dDE$_2$ for benchmark suite of CEC'08.

Functtion	Mean error	
	Dimension 100	Dimension 500
g_1	0.00E+00	0.00E+00
g_2	9.35E+01	6.96E+01
g_3	8.16E+01	8.45E+02
g_4	1.63E+00	1.16E+02
g_5	0.00E+00	0.00E+00
g_6	0.00E+00	0.00E+00
g_7	−1.48E+03	−6.77E+03

Table 8
Average Friedman's rankings with Holm's correction ($\alpha = 0.05$) for CEC'08 functions. Symbol * indicates the Control algorithm. In bold, our proposal's results.

Dimension	Algorithm	Rank	*Holm's A_p*
100	*MTS	2.42	–
	dDE$_2$	**3.71**	**8.99E−01**
	DMS-L-PSO	3.71	8.99E−01
	jDEdynNP-F	3.78	8.99E−01
	LSEDA-gl	4.50	4.54E−01
	DEwSAcc	5.00	2.47E−01
	MLCC	6.14	2.73E−02
	EPUS-PSO	6.71	7.44E−03
500	*MTS	2.85	–
	dDE$_2$	**3.28**	**1.00E+00**
	jDEdynNP-F	3.28	1.00E+00
	LSEDA-gl	4.14	9.78E−01
	DMS-L-PSO	4.85	5.06E−01
	MLCC	5.28	3.18E−01
	DEwSAcc	6.00	9.82E−02
	EPUS-PSO	6.28	6.18E−02

Table 9
Number of best (mean) error values with regards to different functions features when comparing dDE$_2$ versus other algorithms presented in CEC'05 for dimension 30. The results are presented in form of (win, tie, lose).

F./A.	dDE$_2$ versus G-CMA-ES	dDE$_2$ versus BLX-GL50	dDE$_2$ versus L-CMA-ES	dDE$_2$ versus BLXMA	dDE$_2$ versus K-PCX	dDE$_2$ versus SPC-PNX	dDE$_2$versus DE
Separable	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)	(1,0,0)	(0,0,1)
Non-separable	(10,1,8)	(10,1,8)	(11,0,8)	(9,1,9)	(11,0,8)	(10,1,8)	(15,0,4)
Multimodal	(11,1,8)	(11,1,8)	(12,0,8)	(10,1,9)	(12,0,8)	(11,1,8)	(15,0,5)
Rotated	(7,1,7)	(7,1,7)	(8,0,7)	(7,1,7)	(9,0,6)	(6,1,8)	(12,0,3)
Non-rotated	(4,0,1)	(4,0,1)	(4,0,1)	(3,0,2)	(3,0,2)	(5,0,0)	(3,0,2)
Shifted	(4,0,5)	(4,0,5)	(5,0,4)	(3,0,6)	(3,0,6)	(5,0,4)	(4,0,5)
Non-shifted	(7,1,3)	(7,1,3)	(7,0,4)	(7,1,3)	(9,0,2)	(6,1,4)	(11,0,0)
Scalable	(11,1,8)	(11,1,8)	(12,0,8)	(10,1,9)	(12,0,8)	(11,1,8)	(15,0,5)
Hybrid	(9,1,3)	(8,1,4)	(9,0,4)	(8,1,4)	(11,0,2)	(8,1,4)	(13,0,0)
Non-hybrid	(2,0,5)	(3,0,4)	(3,0,4)	(2,0,5)	(1,0,6)	(3,0,4)	(2,0,5)

Table 10
Number of best (mean) error values with regards to different functions features when comparing dDE$_2$ versus other algorithms presented in CEC'05 for dimension 50. The results are presented in form of (win, tie, lose).

F./A.	dDE$_2$ versus G-CMA-ES	dDE$_2$ versus L-CMA-ES
Separable	(1,0,0)	(1,0,0)
Non-separable	(12,0,7)	(12,0,7)
Multi-modal	(13,0,7)	(13,0,7)
Rotated	(9,0,6)	(9,0,6)
Non-rotated	(4,0,1)	(4,0,1)
Shifted	(4,0,5)	(5,0,4)
Non-shifted	(9,0,2)	(8,0,3)
Scalable	(13,0,7)	(13,0,7)
Hybrid	(11,0,2)	(10,0,3)
Non-hybrid	(2,0,5)	(3,0,4)

obtains a higher number of "wins" (better means) on non-separable, multimodal, rotated, scalable, and hybrid functions (as well as in separable, non-rotated, and non-shifted). In fact, these results are repeated when comparing dDE$_2$ versus all other algorithms in this benchmark. We have to notice that, even for rotated functions for which CMA-ES approaches behave specially well on invariant (to rotation) functions [45], our proposal obtains a similar or higher number of "wins" than G-CMA-ES and L-CMA-ES, mostly on dimension 50. In addition, when compared to Canonical DE (right column in Table 9), the number of "wins" reached by dDE$_2$ is higher for practically all function properties, meaning that the induced distribution strategy should be responsible of such an accurate performance, mainly on non-separable and multimodal complex problems. However, on shifted functions, our approach reach moderate results with regards to the compared techniques. As happens with other DE base-line algorithms, dDE$_2$ still shows certain dependence to the origin of coordinates when the optima is far from this point of the search landscape.

Table 11
Number of best (mean) error values with regards to different functions features when comparing dDE_2 versus other algorithms presented in CEC'08 for dimension 100. The results are presented in form of (win, tie, lose).

F./A.	dDE_2 versus MLCC	dDE_2 versus EPUS-PSO	dDE_2 versus jDEdynNP-F	dDE_2 versus MTS	dDE_2 versus DEwSAcc	dDE_2 versus DMS-L-PSO	dDE_2 versus LSEDA-gl
Separable	(2,0,1)	(3,0,0)	(2,0,1)	(0,2,1)	(3,0,0)	(1,2,0)	(3,0,0)
Non-separable	(3,0,1)	(2,0,2)	(2,0,2)	(1,1,2)	(2,0,2)	(1,1,2)	(2,0,2)
Unimodal	(1,0,1)	(1,0,1)	(1,0,1)	(0,1,1)	(1,0,1)	(0,1,1)	(1,0,1)
Multi-modal	(4,0,1)	(4,0,1)	(3,0,2)	(1,2,2)	(4,0,1)	(2,2,1)	(4,0,1)
Non-rotated	(5,0,2)	(5,0,2)	(4,0,3)	(1,3,3)	(5,0,2)	(2,3,2)	(5,0,2)
Shifted	(4,0,2)	(5,0,1)	(4,0,2)	(0,3,3)	(5,0,1)	(2,3,1)	(5,0,1)
Non-shifted	(1,0,0)	(0,0,1)	(0,0,1)	(1,0,0)	(0,0,1)	(0,0,1)	(0,0,1)
Scalable	(5,0,2)	(5,0,2)	(4,0,3)	(1,3,3)	(5,0,2)	(2,3,2)	(5,0,2)

Nevertheless, this weakness of dDE_2 on shifted functions is solved in the context of large scale CEC'08 functions. In this regard, we can observe in Table 11 that our proposal obtains the highest number of "wins" on non-separable, multimodal, non-rotated, scalable, hybrid, and also on shifted functions, in comparison with almost all other algorithms. There is an exception in the case of MTS which constitutes the current state of the art technique for this benchmark, although without statistical differences (compared with dDE_2) as shown in the previous analysis.

In summary, the distribution strategy adopted to dDE_2 seems to be responsible of the successful performance of our proposal on non-separable and multimodal functions, as well as other structural modifications applied to functions like: rotation and shifting. This accurate performance of our approach is also kept in the context scaling functions, since it is always located in the top of compared algorithms for dimensions: 30, 50, 100, and, 500 problem variables.

6. Conclusions

In this work we have experimentally studied, in terms of the quality of solutions, the performance of a two island distributed Differential Evolution whose population is structured in two subpopulations running in parallel, with a certain migration policy. Using the exploitation abilities of DE, and incorporating a diversification mechanism by means of migrant solutions, we can provide it with a higher search capacity in order to improve its global performance. The resulted algorithm (dDE_2) has been thoroughly tested on two standard benchmarks of complex functions (CEC'05 and CEC'08), and has been compared against other competitive approaches in the state of the art.

The main conclusions can be outlined as follows:

1. Our proposal shows a highly competitive performance in comparison with the canonical version of DE and a four islands version of DE (dDE_4). In addition, dDE_2 obtained statistically better results than the island-distribution DE of [9], on which our proposal is based on. Besides, we have even improved the results of relevant literature algorithms using evaluation standard protocols adopted in the benchmark suites of CEC'05 and CEC'08.
2. Concerning the CEC'05 benchmark suite, our model with two islands (dDE_2) is the second best algorithm and statistically equivalent to the best working with dimension 30. However, our proposal beats all the algorithms on dimension 50, showing a competitive performance in comparison with two well-known specialized variants of CMA-ES.
3. In comparison with the algorithms presented in CEC'08, dDE_2 also achieves the second best rank in dimension 100 and the third one in dimension 500. Besides its simplicity, we can notice that our proposal obtains competitive results also when dealing with large scale problems, showing similar performance to highly specialized techniques like MTS or jDE-dynNP-F. Simplicity is important (Occam's Razor), since it has practical effects in allowing an easy parameterization, easy understanding of the behavior of the algorithm and fast implementation of the algorithm.
4. The distribution strategy adopted to dDE_2 seems to be responsible for the successful performance of our proposal on non-separable, multimodal, rotated, and shifted functions. All these features are representative of complex problem characterizations, so we have interested to take them into consideration to test our proposal in this study.
5. The accurate performance of our approach is also kept in the context scaling functions for dimensions: 30, 50, 100, and 500 problem variables, and in the context of standard protocols (CEC'05 and CEC'08). This is hard to see in present research in new algorithms, where scalability is even dismissed and not analyzed in articles.

As a matter of the further work, we are actually working on new hybrid techniques based on Differential Evolution and Particle Swarm methods in order to experiment with different variations of the parallel configurations, as well as the evaluation of new specialized test suites of optimization functions with larger dimensions (CEC'10 and CEC'13).

Acknowledgements

J. Apolloni and G. Leguizamón are supported by PROICO 308083 from Universidad Nacional de San Luis. E. Alba and J. García-Nieto acknowledge funds from the Spanish Ministry of Economy and Competitiveness and FEDER with the roadME project under contract TIN2011-28194 (`http://roadme.lcc.uma.es`). It is also partially founded by project number 8.06/5.47.4142 in collaboration with the VSB-Technical University of Ostrava.

References

[1] T. Bäck, D.B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Computational Intelligence Library, The Institute of Physics, Ringbound, 1997.
[2] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: overview and conceptual comparison, ACM Comput. Surv. 35 (2003) 268–308.
[3] R. Storn, K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359.
[4] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Natural Computing Series, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.
[5] E. Alba, Parallel Metaheuristics: A New Class of Algorithms, Wiley, 2005.
[6] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, S. Tiwari, Problem Definitions and Evaluation Criteria for the CEC'05 Special Session on Real-parameter Optimization, Technical Report KanGAL Report 2005005, Nanyang Technological University, Singapore and Kanpur, India, 2005.
[7] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen, Z. Yang, Benchmark Functions for the CEC'08 Special Session and Competition on Large Scale Global Optimization, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. URL: <http://nical.ustc.edu.cn/cec08ss.php>.
[8] D. Zaharie, D. Petcu, Parallel implementation of multi-population differential evolution, in: Concurrent Information Processing and Computing, 2005, pp. 223–232.
[9] D.K. Tasoulis, N. Pavlidis, V.P. Plagianakos, M.N. Vrahatis, Parallel differential evolution, in: Proceedings of the 2004 IEEE Conference on Congress on Evolutionary Computation, CEC, pp. 2023–2029, 2004.
[10] K.N. Kozlov, A.M. Samsonov, New migration scheme for parallel differential evolution, in: Proceedings the Fifth International Conference on Bioinformatics of Genome Regulation and Structure, vol. 2, pp. 141–144, 2006.
[11] I. De Falco, A. Della Cioppa, D. Maisto, U. Scafuri, E. Tarantino, Satellite image registration by distributed differential evolution, in: Applications of Evolutionary Computing, Lecture Notes in Computer Science, vol. 4448, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 251–260.
[12] I. De Falco, U. Scafuri, E. Tarantino, A. Della Cioppa, A distributed differential evolution approach for mapping in a grid environment, in: Proceedings of the Fifteenth Euromicro International Conference on Parallel, Distributed and Network-based Processing, IEEE Computer Society, 2007, pp. 442–449.
[13] J. Apolloni, G. Leguizamón, J. García-Nieto, E. Alba, Island based distributed differential evolution: an experimental study on hybrid testbeds, in: Proceedings of the Eighth International Conference on Hybrid Intelligent Systems, pp. 696–701, 2008.
[14] D. Izzo, M. Ruciński, C. Ampatzis, Parallel global optimisation meta-heuristics using an asynchronous island-model, in: Proceedings of the 2009 IEEE Congress on Evolutionary Computation, pp. 2301–2308, 2009.
[15] M. Weber, F. Neri, V. Tirronen, Distributed differential evolution with explorative-exploitative population families, Genet. Program. Evolvable Mach. 10 (2009) 343–371.
[16] M. Weber, V. Tirronen, F. Neri, Scale factor inheritance mechanism in distributed differential evolution, Soft Comput. 14 (2010) 1187–1207.
[17] M.F. Tasgetiren, P. Suganthan, Q.-K. Pan, An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem, Appl. Math. Comput. 215 (2010) 3356–3368.
[18] M. Weber, F. Neri, V. Tirronen, Shuffle or update parallel differential evolution for large-scale optimization, Soft Comput. 15 (2011) 2089–2107.
[19] B. Dorronsoro, P. Bouvry, Improving classical and decentralized differential evolution with new mutation operator and population topologies, IEEE Trans. Evol. Comput. 15 (2011) 67–98.
[20] P. Bujok, J. Tvrdík, Parallel migration models applied to competitive differential evolution, in: Swarm and Evolutionary Computation, Lecture Notes in Computer Science, vol. 7269, Springer, Berlin Heidelberg, 2012, pp. 39–47.
[21] D. Xie, L. Ding, X. Du, Y. Hu, S. Wang, Self-adaptive pseudo-parallel differential evolution algorithm, J. Comput. Inf. Syst. 8 (2012) 3403–3411.
[22] M. Biazzini, A. Montresor, Gossiping differential evolution: a decentralized heuristic for function optimization in P2P networks, in: Proceedings of the IEEE Sixteenth International Conference on Parallel and Distributed Systems, pp. 468–475, 2010.
[23] Y. Sun, Y. Li, J. Liu, G. Liu, An improved differential evolution algorithm with ensemble of population topologies, J. Comput. Inf. Syst. 8 (2012) 8667–8674.
[24] J. Nocedal, S. Wright, Numerical Optimization, second ed., Series in Operations Research, Springer, 2006.
[25] D. Molina, F. Herrera, M. Lozano, Adaptive local search parameters for real-coded memetic algorithms, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 888–895, 2005.
[26] C. García-Martínez, M. Lozano, Hybrid real-coded genetic algorithms with female and male differentiation, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 896–903, 2005.
[27] P. Posik, Real-parameter optimization using the mutation step co-evolution, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 872–879, 2005.
[28] J. Ronkkonen, S. Kukkonen, K. Price, Real-parameter optimization with differential evolution, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 506–513, 2005.
[29] A. Auger, N. Hansen, A restart CMA evolution strategy with increasing population size, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1769–1776, 2005.
[30] A. Sinha, S. Tiwari, K. Deb, A population-based, steady-state procedure for real-parameter optimization, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 514–521, 2005.
[31] A. Auger, N. Hansen, Performance evaluation of an advanced local search evolutionary algorithm, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1777–1784, 2005.
[32] P. Ballester, J. Stephenson, J. Carter, K. Gallagher, Real-parameter optimization performance study on the CEC-2005 benchmark with SPC-PNX, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 498–505, 2005.
[33] L.-Y. Tseng, C. Chen, Multiple trajectory search for large scale global optimization, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 3052–3059, 2008.
[34] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 1663–1670, 2008.
[35] J. Brest, A. Zamuda, B. Bošković, M. Maucec, V. Žumer, High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 2032–2039, 2008.
[36] Y. Wang, B. Li, A restart univariate estimation of distribution algorithm: sampling under mixed Gaussian and Lévy probability distribution, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 3917–3924, 2008.
[37] S. Zhao, J. Liang, P. Suganthan, M. Tasgetiren, Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 3845–3852, 2008.
[38] A. Zamuda, J. Brest, B. Bošković, V. Žumer, Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 3718–3725, 2008.
[39] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, S.-J. Tsai, Solving large scale global optimization using improved particle swarm optimizer, in: Proceedings of the 2008 IEEE Congress on Evolutionary Computation, pp. 1777–1784, 2008.
[40] E. Alba, B. Dorronsoro, Cellular Genetics Algorithms, Springer-Velarg, 2008.
[41] E. Alba, MALLBA Group, MALLBA: a library of skeletons for combinatorial optimisation, in: Proceedings of the Euro-Par, LNCS, vol. 2400, pp. 927–932, 2002.

[42] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005, J. Heuristics 15 (2009) 617–644.
[43] D.J. Sheskin, Handbook of Parametric and Nonparametric Statistical Procedures, Chapman & Hall/CRC, 2007.
[44] R. Wilcox, New Statistical Procedures for the Social Sciences, Hillsdale, 1987.
[45] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, A. Auger, Impacts of invariance in search: when CMA-ES and PSO face ill-conditioned and non-separable problems, Appl. Soft Comput. 11 (2011) 5755–5769.