Soft Comput (2011) 15:2157–2174
DOI 10.1007/s00500-010-0644-5

FOCUS

Self-adaptive differential evolution algorithm using population size reduction and three strategies

Janez Brest · Mirjam Sepesy Maučec

Published online: 16 September 2010

Abstract Many real-world optimization problems are large-scale in nature. In order to solve these problems, an optimization algorithm is required that is able to apply a global search regardless of the problems' particularities. This paper proposes a self-adaptive differential evolution algorithm, called jDElscop, for solving large-scale optimization problems with continuous variables. The proposed algorithm employs three strategies and a population size reduction mechanism. The performance of the jDElscop algorithm is evaluated on a set of benchmark problems provided for the Special Issue on the Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems. Non-parametric statistical procedures were performed for multiple comparisons between the proposed algorithm and three well-known algorithms from literature. The results show that the jDElscop algorithm can deal with large-scale continuous optimization effectively. It also behaves significantly better than other three algorithms used in the comparison, in most cases.

Keywords Differential evolution · Self-adaptation · Large-scale optimization · Multiple statistical comparison

1 Introduction

The problem of global optimization naturally arises over many applications, e.g., in advanced engineering design, data analysis, financial planning, risk management, scientific modeling, etc. (Mladenovic et al. 2008). Most cases of practical interest are characterized by multiple local optima. The heuristic methods do not offer a guarantee of locating the global optimum, but give satisfactory results for a wide range of global optimization problems.

Many real-world problems can be formulated as continuous optimization problems (COP). The goal of COP is to find variables of vector $\mathbf{x} = \{x_1, x_2, \ldots, x_D\}$, such that objective function $f(\mathbf{x})$ is optimized. D denotes the dimensionality of the function. The domains of the variables are defined by their lower and upper bounds $[x_{j,\text{low}}, x_{j,\text{upp}}]$ for $j = 1, 2, \ldots, D$.

This paper considers a large scale unconstrained COP, and proposes a self-adaptive differential evolution algorithm jDElscop to solve COPs.

The paper is structured as follows. Section 2 surveys the latest work on large-scale optimization. Section 3 reviews already defined mechanisms of self-adapted differential evolution, which are also used in our new algorithm. Section 4 describes the novel variation of the algorithm, called jDElscop. Section 5 shows the experimental results on the benchmark functions. A comparative statistical analysis of multiple algorithms using non-parametric tests is given and discussed. Section 6 summarizes the important conclusions of this paper.

2 Related work

A differential evolution (DE) algorithm belongs to evolutionary algorithms (EAs). The DE algorithm was proposed by Storn and Price (1997), and since then the DE algorithm has been used in different areas. The original DE was modified, and many new versions proposed in (Price et al.

This work was supported in part by the Slovenian Research Agency under programs P2-0041 and P2-0069.

J. Brest (✉) · M. S. Maučec
Faculty of Electrical Engineering and Computer Science, University of Maribor, Smetanova ulica 17, 2000 Maribor, Slovenia
e-mail: janez.brest@uni-mb.si

2005; Zhang and Sanderson 2009; Das et al. 2009; Rahnamayan et al. 2008; Feoktistov 2006). Neri and Tirronen (2010) give a survey of recent advances in differential evolution.

The original DE algorithm has three control parameters, which are being fixed during the optimization process. Later, various adaptation mechanisms were proposed to overcome the hand-tuning problems of the DE control parameters (Qin et al. 2009; Qin and Suganthan 2005; Brest et al. 2006a; Zhang and Sanderson 2009; Teng et al. 2009).

Teo (2006) proposed a DE algorithm with a dynamic population sizing strategy based on self-adaptation, while F and CR control parameters are also self-adapted. Brest et al. (2007) compared some versions of adaptive and self-adaptive algorithms. Qin and Suganthan in (Qin and Suganthan 2005; Qin et al. 2009) proposed a self-adaptive differential evolution algorithm (SaDE), where the choice of learning strategy and the two control parameters F and CR are not required to be pre-defined. During evolution, the suitable learning strategy and parameter settings are gradually self-adapted, according to the learning experience.

Brest et al. (2006a) proposed a self-adaptive jDE algorithm. The self-adaptive mechanism was used in algorithms for unconstrained optimization (Brest et al. 2006a, 2007; Brest and Maučec 2008). Another version of the jDE algorithm with population size reduction was used for large scale global optimization on CEC 2008 (Brest et al. 2008). Yet another version, based on the jDE algorithm, was used for dynamic optimization problems (Brest et al. 2009). The jDE algorithm was also adopted for the solving of constrained optimization problems (Brest et al. 2006b; Brest 2009). The self-adaptation mechanism proposed in (Brest et al. 2006a) has been widely-used in many practical cases (Caponio et al. 2009) and have shown excellent results. The population size reduction mechanism has a good performance in terms of robustness (Neri and Tirronen 2010).

Historically, scaling EAs to large scale problems has attracted much interest, including both theoretical and practical studies (Tang et al. 2007, 2009; Molina et al. 2009a). To solve high-dimensional problems (MacNish 2007; Tang et al. 2007) cooperative coevolution (Potter and De Jong 1994; Sofge et al. 2002; Zamuda et al. 2008) is often used. Liu et al. (2001) used FEP (fast evolutionary programming) with cooperative coevolution (FEPCC) to speed up convergence rates on large-scale problems; van den Bergh and Engelbrecht (2004) used a cooperative approach to particle swarm optimization (PSO). Yang et al. (2007) used differential evolution with cooperative coevolution (DECC). Molina et al. (2009b); Gao and Wang (2007); Muelas et al. (2009) used a memetic DE algorithm for high-dimensional problem optimization. Other techniques are using local search (Molina et al. 2009a; Muelas et al. 2009; Zhao et al. 2008), neighborhood search (Muelas et al. 2009; Yang et al. 2008c). Estimation of distribution algorithm (EDA) (Wang and Li 2008) is used to tackle those problems.

Recently, Yang, Tang, and Yao proposed DECC-G (Yang et al. 2008a), and MLCC (Yang et al. 2008b) algorithms. Both algorithms use cooperative coevolution for solving large-scale optimization problems. In (Brest et al. 2010) a comparison is presented of experimental results for *jDElsgo*, DECC-G, DECC-G*,[1] and MLCC algorithms.

3 Differential evolution and *jDE* algorithms

This section provides an overview of those algorithms and mechanisms necessary for a better understanding of our new algorithm, as proposed in the next section.

3.1 Differential evolution

The population of the original DE algorithm (Storn and Price 1997) contains NP individuals. An individual is defined as a D-dimensional vector. If G denotes the generation, the population at generation G consists of:

$$\mathbf{x}_i^{(G)} = \left\{ x_{i,1}^{(G)}, x_{i,2}^{(G)}, \ldots, x_{i,D}^{(G)} \right\}, \quad i = 1, 2, \ldots, \mathrm{NP}. \tag{1}$$

During one generation for each vector $\mathbf{x}_i^{(G)}$, DE employs mutation and crossover operations to produce a trial vector:

$$\mathbf{u}_i^{(G)} = \left\{ u_{i,1}^{(G)}, u_{i,2}^{(G)}, \ldots, u_{i,D}^{(G)} \right\}, \quad i = 1, 2, \ldots, \mathrm{NP}. \tag{2}$$

Then a selection operation is used to choose vectors for the next generation $(G+1)$.

The initial population is usually selected uniformly randomly between the lower $x_{j,\mathrm{low}}$ and upper $x_{j,\mathrm{upp}}$ bounds defined for each variable x_j. These bounds are specified according to the nature of the problem.

3.1.1 Mutation operation

Mutation for each population vector $\mathbf{x}_i^{(G)}$ creates a mutant vector $\mathbf{v}_i^{(G)}$:

$$\mathbf{v}_i^{(G)} = \left\{ v_{i,1}^{(G)}, v_{i,2}^{(G)}, \ldots, v_{i,D}^{(G)} \right\}, \quad i = 1, 2, \ldots, \mathrm{NP}. \tag{8}$$

A new mutant vector can be created using one of the mutation strategies. The most useful strategies (Storn and

[1] DECC-G*: the same as DECC-G, except that the grouping structure was used as prior knowledge. The parameter group size was set to $s = 50$. The adaptive weighting strategy of DECC-G was not used.

$$DE/rand/1: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F \cdot (\mathbf{x}_{r_2}^{(G)} - \mathbf{x}_{r_3}^{(G)}) \tag{3}$$

$$DE/best/1: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{best}^{(G)} + F \cdot (\mathbf{x}_{r_1}^{(G)} - \mathbf{x}_{r_2}^{(G)}) \tag{4}$$

$$DE/current\,to\,best/1: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_i^{(G)} + F \cdot (\mathbf{x}_{best}^{(G)} - \mathbf{x}_i^{(G)}) + F \cdot (\mathbf{x}_{r_1}^{(G)} - \mathbf{x}_{r_2}^{(G)}) \tag{5}$$

$$DE/best/2: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{best}^{(G)} + F \cdot (\mathbf{x}_{r_1}^{(G)} - \mathbf{x}_{r_2}^{(G)}) + F \cdot (\mathbf{x}_{r_3}^{(G)} - \mathbf{x}_{r_4}^{(G)}) \tag{6}$$

$$DE/rand/2: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F \cdot (\mathbf{x}_{r_2}^{(G)} - \mathbf{x}_{r_3}^{(G)}) + F \cdot (\mathbf{x}_{r_4}^{(G)} - \mathbf{x}_{r_5}^{(G)}) \tag{7}$$

Fig. 1 The most useful DE's strategies

Price 1997; Price et al. 2005; Feoktistov 2006) are presented in Fig. 1, where the indexes r_d, $d = 1, \ldots, 5$ represent the random and mutually different integers generated within the range [1, NP] and also different from index i. F is a mutation scale factor within the range [0, 2], usually <1. Vector $\mathbf{x}_{\text{best}}^{(G)}$ is the best vector in generation G. Some authors use the name 'target to best' for the strategy in Eq. 5.

3.1.2 Crossover operation

After mutation, a 'binary' crossover operation forms the trial vector $\mathbf{u}_i^{(G)}$ according to the target vector $\mathbf{x}_i^{(G)}$ and its corresponding mutant vector $\mathbf{v}_i^{(G)}$.

$$u_{i,j}^{(G)} = \begin{cases} v_{i,j}^{(G)} & \text{if } \text{rand}(0,1) \le \text{CR or } j = j_{\text{rand}}, \\ x_{i,j}^{(G)} & \text{otherwise} \end{cases} \tag{9}$$

$$i = 1, 2, \ldots, \text{NP and } j = 1, 2, \ldots, D.$$

CR is a crossover control parameter or factor within the range [0, 1) and presents the probability of creating parameters for a trial vector from the mutant vector. Index j_{rand} is a randomly chosen integer within the range [1, NP]. It ensures that the trial vector contains at least one parameter from the mutant vector. Here we have described the binary crossover operation (bin). The other DE crossover operation that could also be used in optimizations is exponential (exp).

If some components of the trial vector are out of bounds, the proposed solutions in literature Storn and Price (1997, 1995), Rönkkönen et al. (2005), Price et al. (2005) are: they are reflected from bounds, set on bounds or used as they are (out of bounds).

3.1.3 Selection operation

The DE algorithm uses a greedy selection. The selection operator selects between the target and corresponding trial vectors. A member of the next generation becomes the fittest vector, i.e., vector with the better fitness value. For example, if we have a minimization problem, we will use the following selection rule:

$$\mathbf{x}_i^{(G+1)} = \begin{cases} \mathbf{u}_i^{(G)} & \text{if } f(\mathbf{u}_i^{(G)}) \le f(\mathbf{x}_i^{(G)}), \\ \mathbf{x}_i^{(G)} & \text{otherwise.} \end{cases} \tag{10}$$

The DE is depicted in Algorithm 1. As can be seen in Algorithm 1, it is population-based algorithm, where three operations: mutation (line 5), crossover (lines 6–13), and selection (lines 14–18) are used to generate the new vector $\mathbf{x}_i^{(G+1)}$.

```
1:  Initialization(); {Generate uniformly distributed random population}
2:  while not termination condition met do
3:    for (i = 0; i < NP; i++) do
4:      Select random indexes r1, r2, and r3 to be different from each
        other and from the index i.
5:      v_i^(G) = x_r1^(G) + F × (x_r2^(G) − x_r3^(G))
6:      j_rand = rand{1, D}
7:      for (j = 0; j < D; j++) do
8:        if (rand(0, 1) ≤ CR or j == j_rand) then
9:          u_ij^(G) = v_ij^(G)
10:       else
11:         u_ij^(G) = x_ij^(G)
12:       end if
13:     end for
14:     if (f(u_i^(G)) ≤ f(x_i^(G))) then
15:       x_i^(G+1) = u_i^(G)
16:     else
17:       x_i^(G+1) = x_i^(G)
18:     end if
19:   end for
20: end while
```

Algorithm 1 Differential evolution algorithm

3.2 Self-adaptive differential evolution

The self-adaptive differential evolution (jDE) algorithm, based on the self-adapting control parameter mechanism, was proposed by (Brest et al. 2006a). The self-adaptive control mechanism is used to change control parameters, i.e., select weighting factor F and crossover constant CR, during the run. The self-adaptive control parameters $F_i^{(G+1)}$ and $\text{CR}_i^{(G+1)}$ are calculated as follows (Brest et al. 2006a):

$$F_i^{(G+1)} = \begin{cases} F_l + \text{rand}_1 \cdot F_u & \text{if } \text{rand}_2 < \tau_1, \\ F_i^{(G)} & \text{otherwise,} \end{cases}$$

$$\text{CR}_i^{(G+1)} = \begin{cases} \text{rand}_3 & \text{if } \text{rand}_4 < \tau_2, \\ \text{CR}_i^{(G)} & \text{otherwise.} \end{cases}$$

They produce control parameters F and CR in a new parent vector. The quantities rand_j, $j \in \{1, 2, 3, 4\}$ represent uniform random values within the range [0, 1]. τ_1 and τ_2 are probabilities to adjust control parameters F and CR, respectively. Constants τ_1, τ_2, F_l, F_u are assigned fixed values to 0.1, 0.1, 0.1, 0.9, respectively. The new F takes value from $[0.1, 1.0]$ and the new CR from [0,1] randomly.

The control parameter values $F_i^{(G+1)}$ and $\mathrm{CR}_i^{(G+1)}$ are obtained before the mutation operation is performed. This means, they influence the mutation, crossover, and selection operations of the new vector $\mathbf{x}_i^{(G+1)}$.

In Brest et al. (2006a) the third control parameter, population size NP, was not changed during the evolutionary process.

Some ideas, on how to improve the jDE algorithm, are reported by Brest et al. (2007), but here we used the original jDE algorithm and mechanisms, as described hereinafter.

3.3 *jDEdynNP-F* algorithm

The *jDEdynNP-F* algorithm (Brest et al. 2008) was used for solving high-dimensional real-parameter optimization problems on CEC 2008. In this subsection we give an overview of the *jDEdynNP-F* algorithm. In this algorithm, F and CR control parameters are self-adapted (as in the *jDE* algorithm), and two mechanisms are used: population size reduction and F sign changing.

3.3.1 *Population size reduction*

In the *jDEdynNP-F* algorithm population size NP was decreased during the evolutionary process. If we assume that we have to optimize a minimization problem then one step of the population size reduction is calculated as follows (Brest and Maučec 2008; Brest et al. 2008):

$$\mathbf{x}_i^{(G)} = \begin{cases} \mathbf{x}_{\frac{NP}{2}+i}^{(G)} & \text{if } f\left(\mathbf{x}_{\frac{NP}{2}+i}^{(G)}\right) < f\left(\mathbf{x}_i^{(G)}\right) \wedge G = G_R, \\ \mathbf{x}_i^{(G)} & \text{otherwise}, \end{cases} \tag{11}$$

$$\mathrm{NP}^{(G+1)} = \begin{cases} \frac{\mathrm{NP}^{(G)}}{2} & \text{if } G = G_R, \\ \mathrm{NP}^{(G)} & \text{otherwise}, \end{cases} \tag{12}$$

for $i = 1, 2, \ldots, \frac{\mathrm{NP}}{2}$.

Note that only a small part of the generations are exposed to reduction. Let G_R be the generation, whose population is to be reduced. The reduction is performed based on the sequence of comparisons between two individuals. One vector (individual) from the first half $\mathbf{x}_i^{(G)}$ of the current population and a corresponding individual from the second half $\mathbf{x}_{\frac{\mathrm{NP}}{2}+i}^{(G)}$ are compared with regard to their fitness values and the better one is placed (as a survivor) at position i in the first half of the current population. The first part of the current population is the population of the next generation. The new population size is equal to half of the latest one.

The number of different population sizes, denoted as pmax, is the parameter of the evolutionary process. For a given value of pmax, pmax − 1 reductions need to be performed. Commonly, a predefined number of evaluations, denoted as Max_FEs, is set as a termination criteria. Using these two parameters (pmax and Max_FEs), and considering that an equal number of evaluations for each population size should be performed, the calculation of generations exposed to reduction is self-evident.

Special care is needed to preserve the minimal requirements of the original DE algorithm population size. Actually, the *DE/rand/1/bin* strategy requires at least four population members (since indexes i, r_1, r_2, and r_3 must be mutually different).

The properties of the population size reduction are:

- it follows the inspiration of the original DE selection operation and
- it does not require any additional operations, e.g., sorting of all individuals according on their fitness value.

```
prob = 0.75;
if (rand(0,1) < prob && f(x_{r2}^{(G)}) > f(x_{r3}^{(G)}))
    F' = −F;  // sign change
else
    F' = F;
F' is used in the mutation operation
```

Fig. 2 The control parameter F sign changing

3.3.2 *Control parameter F sign changing*

The *jDEdynNP-F* algorithm uses a mechanism that changes the sign of the control parameter $F' = -F$ with some probability $(\mathrm{prob} = 0.75)$ when $f(\mathbf{x}_{r_2}^{(G)}) > f(\mathbf{x}_{r_3}^{(G)})$ during the mutation operation as presented in Fig. 2. One can look at the proposed mechanism of sign changing also as swapping indexes r_2 and r_3.

4 The proposed algorithm

This section presents a new jDElscop algorithm for solving large scale COP consisting of:

- three strategies:
  - *DE/rand/1/bin* strategy, called *jDEbin*,
  - *DE/rand/1/exp* strategy, called *jDEexp*, and
  - *DE/best/1/bin strategy*, called *jDEbest*,
- new population size reduction mechanism, and
- control parameter F sign-change mechanism as described in Sect. 3.3.2.

The self-adaptive F and CR control parameters are applied by each strategy.

The structure of the jDElscop is given in Algorithm 2. The algorithm performs the maximal number of iterations on the NP individuals. In each iteration, only one strategy is active and is applied to the mutation, crossover, and selection operations. The strategies and parameter encodings are described in Sect. 4.1.

1: {*pop* ... population}
2: {*imin* ... index of currently best individual}
3: {$\mathbf{x}_i$ ... i-th individual of population}
4: {*MAX_FEs* ... maximum number of function evaluations}
5: {*s* ... one of strategy (*jDEbin*, *jDEexp*, *jDEbest*)}
6: Initialization() {Generate uniformly distributed random population}
7: **for** ($it = 0$; $it < MAX\_FEs$; $it++$) **do**
8: $i = it$ **mod** *NP* {mod ... modulo operation}
9: **if** ($rand(0,1) < 0.1$ **and** ($it > \frac{MAX\_FEs}{2}$)) **then**
10: $s = 1$; {*jDEbest* strategy}
11: **else**
12: **if** ($i < \frac{NP}{2}$) **then**
13: $s = 2$; {*jDEbin* strategy}
14: **else**
15: $s = 3$; {*jDEexp* strategy}
16: **end if**
17: **end if**
18: {Perform one iteration of the jDE using strategy *s* on $\mathbf{x}_i^{(G)}$}
19: {The jDE applies mutation and crossover to generate trial vector $\mathbf{u}^{(G)}$}
20: $\mathbf{u}^{(G)}$ =jDE(*i*, *pop*, *s*)
21: {Fitness evaluation and selection}
22: **if** ($f(\mathbf{u}^{(G)}) \leq f(\mathbf{x}_i^{(G)})$) **then**
23: $\mathbf{x}_i^{(G+1)} = \mathbf{u}^{(G)}$
24: **end if**
25: **end for**

Algorithm 2 jDElscop algorithm

The *jDEbest* strategy is used with a probability of 0.1, when the counter of iterations *it* exceed half of the maximal number of fitness evaluations. The main idea of using low probability for the *jDEbest* strategy is to keep others strategies as the leading strategies during the evolutionary process. Otherwise, the *jDEbin* and *jDEexp* are used, the first one on the first half of NP individuals and the second on the remaining individuals.

1: {*pop* ... population of *NP* individuals}
2: {**c** ... vector of individuals' fitness values}
3: {k_1, k_2, m ... temporary variables}
4: **for** ($i = 0$; $i < \frac{NP}{4}$; $i++$) **do**
5: $k_2 = \frac{NP}{4} + i$
6: **if** $\mathbf{c}[i] < \mathbf{c}[k_2]$ **then**
7: $pop[i] = pop[k_2]$
8: $\mathbf{c}[i] = \mathbf{c}[k_2]$
9: **end if**
10: **end for**
11: **for** ($i = 0$; $i < \frac{NP}{4}$; $i++$) **do**
12: $k_1 = \frac{NP}{2} + i$
13: $k_2 = \frac{3NP}{4} + i$
14: $m = \frac{NP}{4} + i$
15: **if** $\mathbf{c}[k_1] < \mathbf{c}[k_2]$ **then**
16: $pop[m] = pop[k_1]$
17: $\mathbf{c}[m] = \mathbf{c}[k_1]$
18: **else**
19: $pop[m] = pop[k_2]$
20: $\mathbf{c}[m] = \mathbf{c}[k_2]$
21: **end if**
22: **end for**
23: $NP = \lceil \frac{NP}{2} \rceil$

Algorightm 3 Population size reduction when two parts are used

4.1 DE strategies

Our algorithm uses three DE strategies:

1. *jDEbin* (self-adaptive *DE/rand/1/bin*),
2. *jDEexp* (self-adaptive *DE/rand/1/exp*), and
3. *jDEbest* (self-adaptive *DE/best/1/bin*).

In each iteration of the evolutionary process only one strategy is active, as presented in lines 9–17 in Algorithm 2. All strategies have their own self-adaptive control parameters *F* and CR:

$$\mathbf{x}_i^{(G)} = \left(x_{i,1}^{(G)}, x_{i,2}^{(G)}, \ldots, x_{i,D}^{(G)}, F_i^{(G),1}, \mathrm{CR}_i^{(G),1}, \ldots, F_i^{(G),3}, \mathrm{CR}_i^{(G),3}\right)$$

The control parameters of three DE's strategies are encoded into each individual. The first pair of self-adaptive control parameters $F_i^{(G),1}$ and $\mathrm{CR}_i^{(G),1}$ (e.g. $s = 1$) belong to the *jDEbest* strategy, the second pair (e.g. $s = 2$) belong to the *jDEbin* strategy, and the third pair (e.g. $s = 3$) belong to the *jDEexp* strategy.

The self-adaptive control parameters $F_i^{(G+1),s}$ and $\mathrm{CR}_i^{(G+1),s}$ are calculated as follows:

$$F_i^{(G+1),s} = \begin{cases} F_l^s + \mathrm{rand}_1 \cdot F_u^s & \text{if } \mathrm{rand}_2 < \tau_1, \\ F_i^{(G),s} & \text{otherwise,} \end{cases}$$

$$\mathrm{CR}_i^{(G+1),s} = \begin{cases} \mathrm{CR}_l^s + \mathrm{rand}_3 \cdot \mathrm{CR}_u^s & \text{if } \mathrm{rand}_4 < \tau_2, \\ \mathrm{CR}_i^{(G),s} & \text{otherwise.} \end{cases}$$

The values $F_l^s, F_u^s, CR_l^s, CR_u^s$ for $s \in \{1,2,3\}$ can be calculated on the intervals presented in Sect. 5.2. The *jDEbin* strategy uses ranges similar to those proposed in literature (Brest et al. 2006a). The only exception, in this study parameter $F_l^{(2)} = 0.1 + \sqrt{\frac{1.0}{\mathrm{NP}}}$ is used, while in (Brest et al. 2006a) the constant $F_l^{(2)} = 0.1$ was applied. The reason for this is described in Sect. 4.4 in more detail.

We recommend to use of higher values for the *F* and CR control parameters for the *jDEexp* strategy than for the *jDEbin* strategy.

The mechanism of control parameter *F* sign change (see Sect. 3.3.2) is applied in the *jDEbin* and *jDEexp* strategies.

4.2 Population size reduction

Our algorithm proposes a new population size mechanism as presented in Algorithm 3. The *jDEbin* strategy is applied to the first half of NP individuals. Lines 4–10 in Algorithm 3 correspond to the first part where reduction occurs; while lines 11–22 correspond to the second reduction part that belongs to the *jDEexp* strategy. For example, Fig. 3 depicts the population size reduction when pmax = 4, e.g. three reductions are being performed during the evolutionary process.

In the proposed population size reduction mechanism, individuals, which compete with each other during the population reduction, belong to the same strategy. The algorithm for population size reduction is not computationally expensive and is performed only (pmax − 1) times. Therefore, it causes only a slight increase of the computational time.

4.3 Initialization and population size

The initial population is selected uniform randomly between the lower $x_{j,\text{low}}$ and upper $x_{j,\text{upp}}$ bounds defined for each variable x_j. If some components of the trial vector are out of bounds after the mutation operation is applied, then they are set on bounds.

Population size NP seems to be important DE's parameter. Much more researches relating to F and CR control parameters can be found in literature. All three control parameters depend on each other and also the *good* parameter setting is dependable on the nature of the problem that is being optimized. In the experiments that follow, the number of allowed fitness evaluations is predefined to Max_FEs = 5,000D, and the dimension of variables is large. Therefore the small value for population size parameter NP is highly recommended for DE algorithms.

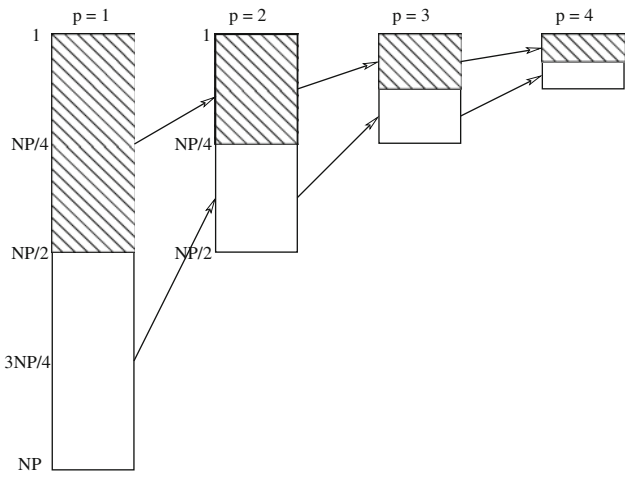

Fig. 3 Population size reduction when two parts are used. The *arrows* show how each part is reduced and its place after reduction

Table 1 Typical run, when Max_FEs=500,000 and pmax = 4 are assumed

p	1	2	3	4
NP_p	100	50	25	13
gen_p	1,250	2,500	5,000	9,615
$\text{NP}_p \times \text{gen}_p$	125,000	125,000	125,000	125,000

In this paper, the population size NP is set to 100, number of reductions is set to pmax = 4, and an equal number of evaluations $\frac{\text{Max\_FEs}}{\text{pmax}}$ for each population size are performed.

An example that is valid for a typical run is illustrated in Table 1, assuming Max_FEs = 500,000, $D = 100$, and pmax = 4. For $\text{NP}_4 = 13$, i.e. $p = \text{pmax}$, the number of generations is equal to $\lfloor \frac{125{,}000}{13} \rfloor = 9{,}615$ and remains 5. The algorithm performs no more than Max_FEs function e.g., fitness evaluations.

Our algorithm optimizes a function as a black box solver for high dimensional problems optimization, however, a well-known cooperative coevolution with any divide-and-conquer strategy is not used.

4.4 Relationship between control parameters

Zaharie (2002) researched about critical values for the control parameters of DE. Premature convergence can be prevented if mutation and crossover induce an increase in the population variance. The values of the parameters which satisfy

$$2F^2 - \frac{2}{\text{NP}} + \frac{\text{CR}}{\text{NP}} = 0 \tag{13}$$

can be considered as critical (Zaharie 2002).

The *jDEbin* strategy uses $\text{CR}_l^{(2)} = 0.0$, which then using Eq. 13 implies that $F_l^{(2)} = \sqrt{\frac{1.0}{\text{NP}}}$. We additionally increase $F_l^{(2)}$ by 0.1.

5 Experiments and results

5.1 Benchmark functions

The jDElscop algorithm was tested on 19 benchmark functions prepared for a Special Issue on the Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems (http://sci2s.ugr.es/eamhco/CFP.php). The benchmark functions are scalable. The dimensions of functions were 50, 100, 200, 500, and 1,000, respectively, and 25 runs of an algorithm were needed for each function. The optimal solution results, $f(\mathbf{x}^*)$, were known for all benchmark functions.

Table 2 Functions F_1–F_{11}

Function	Name	Definition
F_1	Shifted Sphere Function	$\sum_{i=1}^{D} z_i^2 + f\_\text{bias},\ z = x - o$
F_2	Shifted Schwefel Problem 2.21	$\max_i\{\lvert z_i\rvert, 1 \le i \le D\} + f\_\text{bias},\ z = x - o$
F_3	Shifted Rosenbrock's Function	$\sum_{i=1}^{D-1}(100(z_i^2 + z_{i+1})^2 + (z_i - 1)^2) + f\_\text{bias},\ z = x - o$
F_4	Shifted Rastrigin's Function	$\sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f\_\text{bias},\ z = x - o$
F_5	Shifted Griewank's Function	$\sum_{i=1}^{D}\frac{z_i^2}{4{,}000} - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f\_\text{bias},\ z = x - o$
F_6	Shifted Ackley's Function	$-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)) + 20 + e + f\_\text{bias}$
F_7	Shifted Schwefel's Problem 2.22	$\sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$
F_8	Shifted Schwefel's Problem 1.2	$\sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$
F_9	Shifted Extended f_{10}	$\left(\sum_{i=1}^{D-1} f_{10}(x_i, x_{i+1})\right) + f_{10}(x_D, x_1),\ f_{10} = (x^2 + y^2)^{0.25}(\sin^2(50(x^2 + y^2)^{0.1}) + 1)$
F_{10}	Shifted Bohachevsky	$\sum_{i=1}^{D}(x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i) - 0.4\cos(4\pi x_{i+1}) + 0.7)$
F_{11}	Shifted Schaffer	$\sum_{i=1}^{D-1}(x_i^2 + x_{i+1}^2)^{0.25}(\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$

Table 3 Properties of F_1–F_{11}

	Range	Fitness optimum	U/M	Shifted	Separable	Easily optimized dimension by dimension
F_1	$[-100, 100]^D$	−450	U	✓	✓	✓
F_2	$[-100, 100]^D$	−450	U	✓		
F_3	$[-100, 100]^D$	390	M	✓		✓
F_4	$[-5, 5]^D$	−330	M	✓	✓	✓
F_5	$[-600, 600]^D$	−180	M	✓		
F_6	$[-32, 32]^D$	−140	M	✓	✓	✓
F_7	$[-10, 10]^D$	0	U	✓	✓	✓
F_8	$[-65.536, 65.536]^D$	0	U	✓		
F_9	$[-100, 100]^D$	0	U	✓		✓
F_{10}	$[-15, 15]^D$	0	U	✓		
F_{11}	$[-100, 100]^D$	0	U	✓		✓

U/M unimodal/multimodal

Properties of these functions are presented in Tables 2, 3, and 4. Note, the separability of a function presents a measure of difficulty when solving it. However, a function $f(\mathbf{x})$ is separable if its parameters x_i are independent. In general, separable problems are considered as easiest, while the fully nonseparable are usually the most difficult problems to solve (Tang et al. 2009). Although some of the used functions were separable in their original form, applying techniques such as Salomon's random coordinate rotation technique, makes them non-separable. In addition, the global optimum can be shifted.

5.2 Parameter settings

In the experiments, control parameters of the jDElscop algorithm were set as follows:

1. NP was reducible in a deterministic way, initial value $\text{NP}_1 = 100$,
2. pmax was fixed during the optimization, pmax = 4.
3. *jDEbin* strategy:
   - F was self-adaptive on interval $[0.1 + \sqrt{\frac{1.0}{\text{NP}}}, 1.0]$, initially set to 0.5,
   - CR was self-adaptive on interval [0.0, 1.0], initially set to 0.9,
4. *jDEexp* strategy:
   - F was self-adaptive on interval [0.5, 1.0], initially set to 0.5,
   - CR was self-adaptive on interval [0.3, 1.0], initially set to 0.9,

Table 4 Properties of F_{12}–$F_{19}*$

	F_{ns}	F'	m_{ns}	Range	$f(\mathbf{x}^*)$
F_{12}	NS-F_9	F_1	0.25	$[-100, 100]^D$	0
F_{13}	NS-F_9	F_3	0.25	$[-100,100]^D$	0
F_{14}	NS-F_9	F_4	0.25	$[-5, 5]^D$	0
F_{15}	NS-F_{10}	NS-F_7	0.25	$[-10, 10]^D$	0
$F_{16}*$	NS-F_9	F_1	0.5	$[-100, 100]^D$	0
$F_{17}*$	NS-F_9	F_3	0.75	$[-100, 100]^D$	0
$F_{18}*$	NS-F_9	F_4	0.75	$[-5, 5]^D$	0
$F_{19}*$	NS-F_{10}	NS-F_7	0.75	$[-10, 10]^D$	0

NS non-shifted

5. *jDEbest* strategy:

- F was self-adaptive on interval [0.4, 1.0], initially set to 0.5,
- CR was self-adaptive on interval [0.7, 0.95], initially set to 0.9,
- this strategy is used with probability of 0.1 when the counter of iterations *it* exceed half of the maximal number of fitness evaluations.

PC configure:

System: GNU/Linux CPU: 2.5 GHz RAM: 4 GB
Language: C/C++ Algorithm: jDElscop
Compiler: GNU Compiler

5.3 Experimental results

The obtained results [error values $f(\mathbf{x}) - f(\mathbf{x}^*)$] for dimensions $D = 50, 100, 200, 500,$ and 1,000 are presented in Tables 5, 6, 7, 8, and 9, respectively. 25 runs were performed for each function. The best obtained result at the end of each optimization process is recorded during each run. The best, median, worst, mean values and standard deviations are shown in Tables 5, 6, 7, 8, and 9. As suggested for this special issue, we approximate all values below 1e − 14 to 0.0.

Note that the obtained median value is much better than the corresponding mean value, which means that some outliers exists with a worse fitness value but, on average, the algorithm found a solution with good fitness value within the prescribed Max_FEs. For example, the obtained median value in Table 9 for dimension $D = 1{,}000$ is equal to zero in ten cases, lower than 10^{-6} in four cases, and it is worse than 10^{-6} in five cases.

The jDElscop algorithm seems to have difficulties when solving functions F_8 (Shifted Schwefel's Problem 1.2) and F_3 (Shifted Rosenbrock's Function) and hybrid composition functions F_{13} and F_{17} where F_3 is involved.

Tables 10, 11 ,12, 13, and 14 show the average running time in seconds on the 25 runs of the jDElscop algorithm. We used power regression $y = ax^b$ to analyze the time complexity of our algorithm. In our case x is dimensionality D, and y are the times in Tables 10, 11 ,12, 13, and 14. The results of power regression are presented in Table 15. It can be seen that our algorithm has a time complexity lower than $O(D^2)$.

Table 5 Experimental results with dimension $D = 50$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Best	0.00e+00	6.95e−03	1.41e+01	0.00e+00	0.00e+00	5.68e−14	0.00e+00
Median	0.00e+00	2.44e−02	1.86e+01	0.00e+00	0.00e+00	8.53e−14	0.00e+00
Worst	0.00e+00	7.66e−02	7.60e+01	0.00e+00	0.00e+00	1.42e−13	0.00e+00
Mean	0.00e+00	3.15e−02	2.28e+01	0.00e+00	0.00e+00	9.55e−14	0.00e+00
Std	0.00e+00	2.09e−02	1.60e+01	0.00e+00	0.00e+00	2.58e−14	0.00e+00
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Best	2.10e−04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.09e+01	0.00e+00
Median	5.76e−03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.36e+01	0.00e+00
Worst	5.99e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.80e+01	0.00e+00
Mean	9.97e−03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.36e+01	0.00e+00
Std	1.29e−02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.48e+00	0.00e+00
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Best	0.00e+00	0.00e+00	2.51e−06	0.00e+00	0.00e+00		
Median	0.00e+00	0.00e+00	1.07e−02	2.09e−14	0.00e+00		
Worst	0.00e+00	0.00e+00	4.12e−02	7.70e−14	0.00e+00		
Mean	0.00e+00	0.00e+00	7.43e−03	2.41e−14	0.00e+00		
Std	0.00e+00	0.00e+00	9.40e−03	1.60e−14	0.00e+00		

Table 6 Experimental results with dimension $D = 100$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Best	0.00e+00	7.34e−01	5.33e+01	0.00e+00	0.00e+00	1.42e−13	0.00e+00
Median	0.00e+00	1.13e+00	5.60e+01	0.00e+00	0.00e+00	1.99e−13	0.00e+00
Worst	0.00e+00	2.17e+00	1.17e+02	0.00e+00	0.00e+00	2.84e−13	0.00e+00
Mean	0.00e+00	1.21e+00	6.13e+01	0.00e+00	0.00e+00	2.00e−13	0.00e+00
Std	0.00e+00	3.80e−01	1.65e+01	0.00e+00	0.00e+00	3.33e−14	0.00e+00
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Best	1.50e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.99e+01	0.00e+00
Median	3.54e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	4.33e+01	0.00e+00
Worst	3.14e+01	1.20e−07	0.00e+00	1.20e−07	0.00e+00	9.87e+01	0.00e+00
Mean	5.57e+00	7.18e−09	0.00e+00	8.17e−09	0.00e+00	5.11e+01	0.00e+00
Std	5.98e+00	2.63e−08	0.00e+00	2.87e−08	0.00e+00	2.02e+01	0.00e+00
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Best	0.00e+00	0.00e+00	1.51e−04	2.10e−14	0.00e+00		
Median	0.00e+00	0.00e+00	4.30e−02	5.03e−14	0.00e+00		
Worst	0.00e+00	0.00e+00	2.78e+00	2.41e−13	0.00e+00		
Mean	0.00e+00	0.00e+00	3.21e−01	6.33e−14	0.00e+00		
Std	0.00e+00	0.00e+00	7.88e−01	4.48e−14	0.00e+00		

Table 7 Experimental results with dimension $D = 200$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Best	0.00e+00	5.88e+00	1.31e+02	0.00e+00	0.00e+00	3.13e−13	0.00e+00
Median	0.00e+00	7.49e+00	1.36e+02	0.00e+00	0.00e+00	4.55e−13	0.00e+00
Worst	0.00e+00	9.38e+00	1.99e+02	0.00e+00	0.00e+00	5.97e−13	0.00e+00
Mean	0.00e+00	7.54e+00	1.40e+02	0.00e+00	0.00e+00	4.52e−13	0.00e+00
Std	0.00e+00	8.27e−01	1.71e+01	0.00e+00	0.00e+00	7.05e−14	0.00e+00
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Best	9.10e+01	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.01e+02	0.00e+00
Median	2.32e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.04e+02	0.00e+00
Worst	8.58e+02	1.70e−07	0.00e+00	1.20e−07	0.00e+00	1.60e+02	1.03e−14
Mean	2.52e+02	4.30e−08	0.00e+00	9.58e−09	0.00e+00	1.10e+02	4.11e−16
Std	1.61e+02	6.71e−08	0.00e+00	3.31e−08	0.00e+00	1.82e+01	2.05e−15
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Best	0.00e+00	0.00e+00	9.16e+00	1.05e−13	0.00e+00		
Median	0.00e+00	0.00e+00	1.29e+01	1.84e−13	0.00e+00		
Worst	0.00e+00	0.00e+00	7.31e+01	4.60e−13	0.00e+00		
Mean	0.00e+00	0.00e+00	2.39e+01	2.04e−13	0.00e+00		
Std	0.00e+00	0.00e+00	2.32e+01	8.53e−14	0.00e+00		

5.4 Comparison of different algorithms

We compared our jDElscop algorithm with the following algorithms:

- the original DE algorithm with *DE/rand/1/exp* strategy, $F = 0.5$, and CR $= 0.9$, (Storn and Price 1997)
- the real-coded CHC (Eshelman and Schaffer 1993) algorithm, and

Table 8 Experimental results with dimension $D = 500$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Best	0.00e+00	2.65e+01	3.88e+02	0.00e+00	0.00e+00	9.66e−13	0.00e+00
Median	0.00e+00	3.06e+01	3.95e+02	0.00e+00	0.00e+00	1.17e−12	0.00e+00
Worst	0.00e+00	3.80e+01	4.56e+02	9.97e−01	0.00e+00	1.42e−12	0.00e+00
Mean	0.00e+00	3.06e+01	4.06e+02	1.59e−01	0.00e+00	1.18e−12	0.00e+00
Std	0.00e+00	2.52e+00	2.44e+01	3.72e−01	0.00e+00	1.18e−13	0.00e+00
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Best	3.89e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	2.94e+02	0.00e+00
Median	5.65e+03	0.00e+00	0.00e+00	0.00e+00	0.00e+00	3.00e+02	0.00e+00
Worst	8.10e+03	3.39e−07	0.00e+00	2.89e−07	0.00e+00	3.53e+02	9.95e−01
Mean	5.66e+03	6.10e−08	0.00e+00	4.40e−08	0.00e+00	3.14e+02	8.00e−02
Std	1.28e+03	1.08e−07	0.00e+00	7.84e−08	0.00e+00	2.36e+01	2.75e−01
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Best	0.00e+00	0.00e+00	6.29e+01	5.30e−13	0.00e+00		
Median	0.00e+00	0.00e+00	6.64e+01	9.50e−13	0.00e+00		
Worst	0.00e+00	0.00e+00	1.22e+02	2.54e−12	0.00e+00		
Mean	0.00e+00	0.00e+00	7.65e+01	1.11e−12	0.00e+00		
Std	0.00e+00	0.00e+00	2.19e+01	4.85e−13	0.00e+00		

Table 9 Experimental results with dimension $D = 1{,}000$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Best	0.00e+00	5.37e+01	8.36e+02	0.00e+00	0.00e+00	2.25e−12	0.00e+00
Median	0.00e+00	6.21e+01	8.44e+02	0.00e+00	0.00e+00	2.61e−12	0.00e+00
Worst	0.00e+00	6.98e+01	9.00e+02	1.99e+00	0.00e+00	3.47e−12	0.00e+00
Mean	0.00e+00	6.14e+01	8.48e+02	1.99e−01	0.00e+00	2.67e−12	0.00e+00
Std	0.00e+00	4.05e+00	1.57e+01	4.97e−01	0.00e+00	3.24e−13	0.00e+00
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Best	2.41e+04	0.00e+00	0.00e+00	0.00e+00	0.00e+00	6.33e+02	0.00e+00
Median	3.15e+04	5.97e−08	0.00e+00	1.20e−07	0.00e+00	6.41e+02	0.00e+00
Worst	5.17e+04	1.10e−01	0.00e+00	1.07e−02	0.00e+00	6.86e+02	9.95e−01
Mean	3.21e+04	4.40e−03	0.00e+00	8.58e−04	0.00e+00	6.57e+02	3.98e−02
Std	5.49e+03	2.20e−02	0.00e+00	2.97e−03	0.00e+00	2.30e+01	1.99e−01
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Best	0.00e+00	0.00e+00	1.56e+02	1.65e−12	0.00e+00		
Median	0.00e+00	0.00e+00	1.67e+02	2.84e−12	0.00e+00		
Worst	0.00e+00	2.01e+01	2.18e+02	1.14e+00	0.00e+00		
Mean	0.00e+00	8.04e−01	1.72e+02	1.65e−01	0.00e+00		
Std	0.00e+00	4.02e+00	1.67e+01	3.86e−01	0.00e+00		

- the G-CMA-ES (Auger and Hansen 2005) algorithm.

The average error values of 25 runs for each algorithm are shown in Table 16. These are results for all dimensions (50, 100, 200, 500, 1,000) for the original DE, CHC, and jDElscop algorithms, while for the G-CMA-ES algorithm the results are given for dimensions 50, 100, 200, and 500. The results for the original DE, CHC, G-CMA-ES algorithms were prepared for comparison purposes and were

Table 10 Execution times with dimension $D = 50$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Mean	5.02e−01	6.15e−01	5.75e−01	1.07e+00	1.28e+00	9.05e−01	5.19e−01
Std	2.39e−03	2.22e−03	2.01e−03	1.59e−02	6.85e−03	4.45e−03	2.88e−03
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Mean	5.11e−01	3.99e+00	1.46e+00	3.90e+00	1.54e+00	1.64e+00	1.99e+00
Std	1.94e−03	2.17e−02	6.80e−03	1.50e−02	3.90e−02	5.19e−03	2.24e−02
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Mean	8.90e−01	2.51e+00	3.62e+00	3.68e+00	1.29e+00		
Std	2.71e−03	4.24e−02	1.74e−02	4.03e−02	4.45e−03		

Table 11 Execution times with dimension $D = 100$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Mean	1.65e+00	2.05e+00	1.87e+00	3.96e+00	4.67e+00	3.19e+00	1.71e+00
Std	1.10e−02	5.71e−03	6.18e−03	7.68e−02	2.12e−02	1.27e−02	1.09e−02
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Mean	1.66e+00	1.56e+01	5.50e+00	1.54e+01	5.76e+00	6.13e+00	7.59e+00
Std	5.60e−03	7.42e−02	1.56e−02	5.94e−02	1.23e−01	2.06e−02	9.09e−02
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Mean	3.04e+00	9.48e+00	1.37e+01	1.40e+01	4.63e+00		
Std	8.00e−03	1.06e−01	3.60e−02	1.84e−01	1.52e−02		

Table 12 Execution times with dimension $D = 200$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Mean	5.85e+00	7.32e+00	6.72e+00	1.49e+01	1.75e+01	1.19e+01	6.07e+00
Std	3.33e−02	2.39e−02	1.76e−02	2.48e−01	7.65e−02	5.48e−02	3.65e−02
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Mean	5.88e+00	6.19e+01	2.15e+01	6.12e+01	2.22e+01	2.33e+01	2.98e+01
Std	1.65e−02	6.43e−01	2.15e−01	2.19e−01	3.62e−01	5.34e−02	2.85e−01
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Mean	1.13e+01	3.73e+01	5.40e+01	5.51e+01	1.79e+01		
Std	4.06e−02	4.24e−01	1.63e−01	3.83e−01	5.23e−02		

taken from http://sci2s.ugr.es/eamhco/.[2] The obtained results show that our algorithm performed well against all benchmark functions for all dimensions. The jDElscop algorithm found global optimum, i.e., the error value is zero, in many cases, also when $D = 1{,}000$.

[2] It is interesting to note that the results obtained on the test suite by using *DE/rand/1/exp* strategy are clearly better than those obtained by employing the *DE/rand/1/bin* strategy.

Statistical analyses introduced by Demšar (2006), García and Herrera (2008), García et al. (2009), and García et al. (2010) were used to analyze the obtained results of different algorithms.

Under the null hypothesis, the k algorithms are equivalent. If the null hypothesis is rejected, then at least one of the k algorithms performed significant differently from at least one other algorithm. However, this does not indicate which one. A post-hoc test needs to be done to obtain this information.

Table 13 Execution times with dimension $D = 500$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Mean	3.40e+01	4.20e+01	3.89e+01	9.05e+01	1.03e+02	7.11e+01	3.50e+01
Std	3.48e−01	2.28e−01	1.10e−01	1.53e+00	3.38e−01	2.59e−01	2.17e−01
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Mean	3.36e+01	3.82e+02	1.30e+02	3.80e+02	1.35e+02	1.41e+02	1.80e+02
Std	8.46e−02	1.10e+00	3.27e−01	1.15e+00	1.32e+00	3.32e−01	1.43e+00
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Mean	6.89e+01	2.29e+02	3.31e+02	3.39e+02	1.12e+02		
Std	2.70e−01	1.90e+00	9.52e−01	1.82e+00	2.08e−01		

Table 14 Execution times with dimension $D = 1{,}000$

	F_1	F_2	F_3	F_4	F_5	F_6	F_7
Mean	1.33e+02	1.60e+02	1.56e+02	3.58e+02	4.01e+02	2.80e+02	1.40e+02
Std	1.12e+00	1.55e+00	3.40e−01	3.69e+00	1.16e+00	2.07e+00	8.59e−01
	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
Mean	1.33e+02	1.52e+03	5.15e+02	1.52e+03	5.35e+02	5.55e+02	7.15e+02
Std	3.08e−01	3.75e+00	2.91e+00	3.59e+00	4.44e+00	1.55e+00	4.26e+00
	F_{15}	$F_{16}*$	$F_{17}*$	$F_{18}*$	$F_{19}*$		
Mean	2.69e+02	9.06e+02	1.30e+03	1.38e+03	4.33e+02		
Std	7.01e−01	4.19e+00	2.73e+00	4.54e+00	9.61e−01		

Performance comparisons of multiple algorithms can be arranged after ranking the algorithms according to their mean value of each benchmark function. Tables 17, 18, 19, 20, 21 show the average ranking of the algorithms when the dimension varies from 50 through 1,000. Regardless of dimensions, the algorithms can be sorted by average ranking into the following order: jDElscop, DE, G-CMA-ES, and CHC. The best average ranking was obtained by the jDElscop algorithm, which outperformed the other three algorithms.

Tables 17, 18, 19, 20, and 21 show the average ranking for the jDElscop, CHC, and G-CMA-ES algorithms (without the G-CMA-ES algorithm) on a set of all dimensions. The best average ranking was obtained by the jDElscop algorithm, which outperformed the other algorithms.

5.4.1 Comparisons with a control algorithm

The Bonferroni–Dunn test (Dunn 1961) was used for the post-hoc test to detect significant difference for the jDElscop algorithm.

The performance of the jDElscop algorithm is significantly different if the corresponding ranks differ by at least the critical distance *CD* that is defined as follows:

Table 15 Time complexity analysis using power regression $y = ax^b$

Fun.	a	b	rss
F_1	3.4003E−4	1.8570	89.0667
F_2	4.2207E−4	1.8545	56.4348
F_3	3.6457E−4	1.8695	124.0883
F_4	5.3578E−4	1.9402	92.9612
F_5	6.7970E−4	1.9225	144.3762
F_6	4.9230E−4	1.9145	104.7618
F_7	3.7420E−4	1.8837	15.8602
F_8	2.8459E−4	1.8815	280.8795
F_9	2.3326E−3	1.9474	94.2575
F_{10}	5.8788E−4	1.9635	9.4240
F_{11}	1.9603E−3	1.9840	39.4422
F_{12}	7.3089E−4	1.9577	101.7376
F_{13}	8.0106E−4	1.9510	201.2616
F_{14}	9.2232E−4	1.9657	195.8741
F_{15}	4.4827E−4	1.9206	66.9233
$F_{16}*$	1.1245E−3	1.9730	253.3440
$F_{17}*$	1.6323E−3	1.9732	556.6561
$F_{18}*$	1.6087E−3	1.9785	525.2190
$F_{19}*$	5.7719E−4	1.9479	204.5267

rss residual sum of squares

Table 16 Average values of experimental results

	D	jDElscop	DE	CHC	G-CMA-ES
F_1	50	**0.00e+00**	**0.00E+00**	1.67E−11	**0.00E+00**
	100	**0.00e+00**	**0.00E+00**	3.56E−11	**0.00E+00**
	200	**0.00e+00**	**0.00E+00**	8.34E−01	**0.00E+00**
	500	**0.00e+00**	**0.00E+00**	2.84E−12	**0.00E+00**
	1,000	**0.00e+00**	**0.00E+00**	1.36E−11	
F_2	50	3.15e−02	3.60E−01	6.19E+01	**2.75E−11**
	100	1.21e+00	4.45E+00	8.58E+01	**1.51E−10**
	200	7.54e+00	1.92E+01	1.03E+02	**1.16E−09**
	500	3.06e+01	5.35E+01	1.29E+02	**3.48E−04**
	1,000	**6.14e+01**	8.46E+01	1.44E+02	
F_3	50	2.28e+01	2.89E+01	1.25E+06	**7.97E−01**
	100	6.13e+01	8.01E+01	4.19E+06	**3.88E+00**
	200	1.40e+02	1.78E+02	2.01E+07	**8.91E+01**
	500	4.06e+02	4.76E+02	1.14E+06	**3.58E+02**
	1,000	**8.48e+02**	9.69E+02	8.75E+03	
F_4	50	**0.00e+00**	3.98E−02	7.43E+01	1.05E+02
	100	**0.00e+00**	1.27E−01	5.40E+02	6.48E+02
	500	**1.59e−01**	3.20E−01	1.91E+03	2.10E+03
	1,000	**1.99e−01**	1.44E+00	4.76E+03	
F_5	50	**0.00e+00**	**0.00E+00**	1.67E−03	2.96E−04
	100	**0.00e+00**	**0.00E+00**	3.83E−03	1.58E−03
	200	**0.00e+00**	**0.00E+00**	8.76E−03	**0.00E+00**
	500	**0.00e+00**	**0.00E+00**	6.98E−03	2.96E−04
	1,000	**0.00e+00**	**0.00E+00**	7.02E−03	
F_6	50	**9.55e−14**	1.43E−13	6.15E−07	2.09E+01
	100	**2.00e−13**	3.10E−13	4.10E−07	2.12E+01
	200	**4.52e−13**	6.54E−13	1.23E+00	2.14E+01
	500	**1.18e−12**	1.65E−12	5.16E+00	2.15E+01
	1,000	**2.67e−12**	3.29E−12	1.38E+01	
F_7	50	**0.00e+00**	**0.00E+00**	2.66E−09	1.01E−10
	100	**0.00e+00**	**0.00E+00**	1.40E−02	4.22E−04
	200	**0.00e+00**	**0.00E+00**	2.59E−01	1.17E−01
	500	**0.00e+00**	**0.00E+00**	1.27E−01	7.21E+153
	1,000	**0.00e+00**	**0.00E+00**	3.52E−01	
F_8	50	9.97e−03	3.44E+00	2.24E+02	**0.00E+00**
	100	5.57e+00	3.69E+02	1.69E+03	**0.00E+00**
	200	2.52e+02	5.53E+03	9.38E+03	**0.00E+00**
	500	5.66e+03	6.09E+04	7.22E+04	**2.36E−06**
	1,000	**3.21e+04**	2.46E+05	3.11E+05	
F_9	50	**0.00E+00**	2.73E+02	3.10E+02	1.66E+01
	100	**7.18e−09**	5.06E+02	5.86E+02	1.02E+02
	200	**6.10e−08**	2.52E+03	3.00E+03	1.74E+03
	1,000	**4.40e−03**	5.13E+03	6.11E+03	
F_{10}	50	**0.00e+00**	**0.00E+00**	7.30E+00	6.81E+00
	100	**0.00e+00**	**0.00E+00**	3.30E+01	1.66E+01
	200	**0.00e+00**	**0.00E+00**	7.13E+01	4.43E+01
	500	**0.00e+00**	**0.00E+00**	1.86E+02	1.27E+02
	1,000	**0.00e+00**	**0.00E+00**	3.83E+02	
F_{11}	50	**0.00e+00**	6.23E−05	2.16E+00	3.01E+01
	100	**8.17e−09**	1.28E−04	7.32E+01	1.64E+02
	200	**9.58e−09**	2.62E−04	3.85E+02	8.03E+02
	500	**4.40e−08**	6.76E−04	1.81E+03	4.16E+03
	1,000	**8.58e−04**	1.35E−03	4.82E+03	
F_{12}	50	**0.00e+00**	5.35E−13	9.57E−01	1.88E+02
	100	**0.00e+00**	5.99E−11	1.03E+01	4.17E+02
	200	**0.00e+00**	9.76E−10	7.44E+01	9.06E+02
	500	**0.00e+00**	7.07E−09	4.48E+02	2.58E+03
	1,000	**0.00e+00**	1.68E−08	1.05E+03	
F_{13}	50	**1.36e+01**	2.45E+01	2.08E+06	1.97E+02
	100	**5.11e+01**	6.17E+01	2.70E+06	4.21E+02
	200	**1.10e+02**	1.36E+02	5.75E+06	9.43E+02
	500	**3.14e+02**	3.59E+02	3.22E+07	2.87E+03
	1,000	**6.57e+02**	7.30E+02	6.66E+07	
F_{14}	50	**0.00e+00**	4.16E−08	6.17E+01	1.09E+02
	100	**0.00e+00**	4.79E−02	1.66E+02	2.55E+02
	200	**4.11e−16**	1.38E−01	4.29E+02	6.09E+02
	500	**8.00e−02**	1.35E−01	1.46E+03	1.95E+03
	1,000	**3.98e−02**	6.90E−01	3.62E+03	
F_{15}	50	**0.00e+00**	**0.00E+00**	3.98E−01	9.79E−04
	100	**0.00e+00**	**0.00E+00**	8.13E+00	6.30E−01
	200	**0.00e+00**	**0.00E+00**	2.14E+01	1.75E+00
	500	**0.00e+00**	**0.00E+00**	6.01E+01	2.82E+262
	1,000	**0.00e+00**	**0.00E+00**	8.37E+01	
$F_{16}*$	50	**0.00e+00**	1.56E−09	2.95E−09	4.27E+02
	100	**0.00e+00**	3.58E−09	2.23E+01	8.59E+02
	200	**0.00e+00**	7.46E−09	1.60E+02	1.92E+03
	500	**0.00e+00**	2.04E−08	9.55E+02	5.45E+03
	1,000	8.04e−01	**4.18E−08**	2.32E+03	
$F_{17}*$	50	**7.43e−03**	7.98E−01	2.26E+04	6.89E+02
	100	**3.21e−01**	1.23E+01	1.47E+05	1.51E+03
	200	**2.39e+01**	3.70E+01	1.75E+05	3.36E+03
	500	**7.65e+01**	1.11E+02	8.40E+05	9.59E+03
	1,000	**1.72e+02**	2.36E+02	2.04E+07	
$F_{18}*$	50	**2.41e−14**	1.22E−04	1.58E+01	1.31E+02
	100	**6.33e−14**	2.98E−04	7.00E+01	3.07E+02
	200	**2.04e−13**	4.73E−04	2.12E+02	6.89E+02
	500	**1.11e−12**	1.22E−03	7.32E+02	2.05E+03
	1,000	1.65e−01	**2.37E−03**	1.72E+03	
$F_{19}*$	50	**0.00e+00**	**0.00E+00**	3.59E+02	4.76E+00
	100	**0.00e+00**	**0.00E+00**	5.45E+02	2.02E+01
	200	**0.00e+00**	**0.00E+00**	2.06E+03	7.52E+02
	500	**0.00e+00**	**0.00E+00**	1.76E+03	2.44E+06
	1,000	**0.00e+00**	**0.00E+00**	4.20E+03	

The best values are in boldface

Table 17 Average rankings of the algorithms when $D = 50$

Algorithm	Ranking
jDElscop	3.6578947368421035
DE	2.9210526315789473
CHC	1.368421052631579
G-CMA-ES	2.052631578947368

Table 18 Average rankings of the algorithms when $D = 100$

Algorithm	Ranking
jDElscop	3.6578947368421035
DE	2.9210526315789473
CHC	1.368421052631579
G-CMA-ES	2.052631578947368

Table 19 Average rankings of the algorithms when $D = 200$

Algorithm	Ranking
jDElscop	3.6315789473684195
DE	2.894736842105263
CHC	1.368421052631579
G-CMA-ES	2.1052631578947367

Table 20 Average rankings of the algorithms when $D = 500$

Algorithm	Ranking
jDElscop	3.6578947368421035
DE	2.9210526315789473
CHC	1.5263157894736836
G-CMA-ES	1.8947368421052628

Table 21 Average rankings of the algorithms when $D = 1{,}000$

Algorithm	Ranking
jDElscop	2.736842105263158
DE	2.263157894736842
CHC	0.9999999999999996

$$\mathrm{CD} = Q_\alpha \sqrt{\frac{k(k+1)}{6N}},$$

where Q_α is the critical value for a multiple non-parametric comparison with a control (Zar 1999). Four algorithms ($k = 4$) are used in statistical analysis for dimensions $D = 50, 100, 200$, and 500, while $k = 3$ for $D = 1{,}000$, and $N = 19$ denotes the number of functions. Critical difference represents the threshold for the best performing algorithm, i.e., that with the highest ranking. Those rankings which do not exceed the threshold are associated with an algorithm with worse performance than the best performing algorithm.

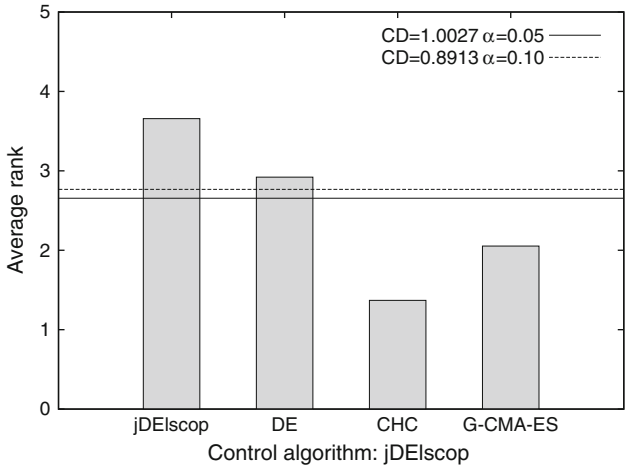

Fig. 4 Bonferroni–Dunn's graphic corresponding to the results when $D = 50$

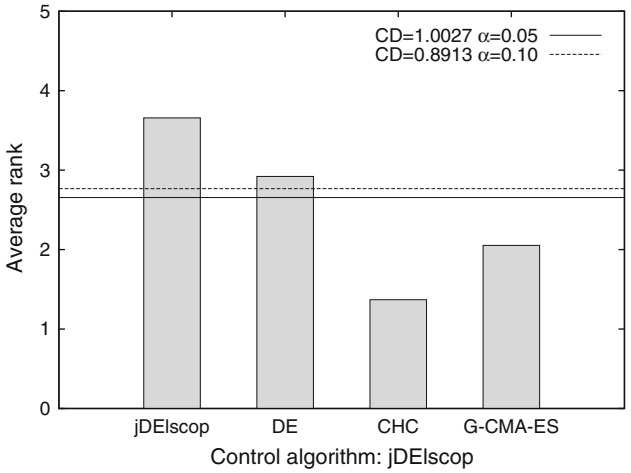

Fig. 5 Bonferroni–Dunn's graphic corresponding to the results when $D = 100$

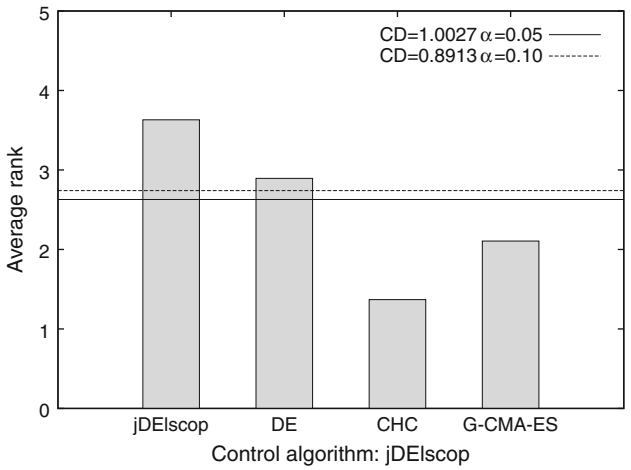

Fig. 6 Bonferroni–Dunn's graphic corresponding to the results when $D = 200$

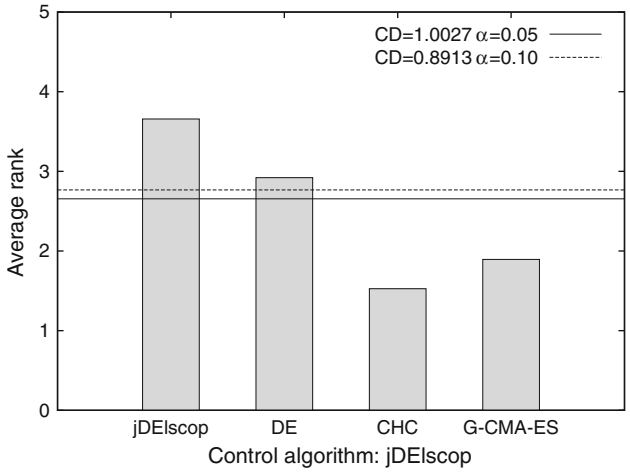

Fig. 7 Bonferroni–Dunn's graphic corresponding to the results when $D = 500$

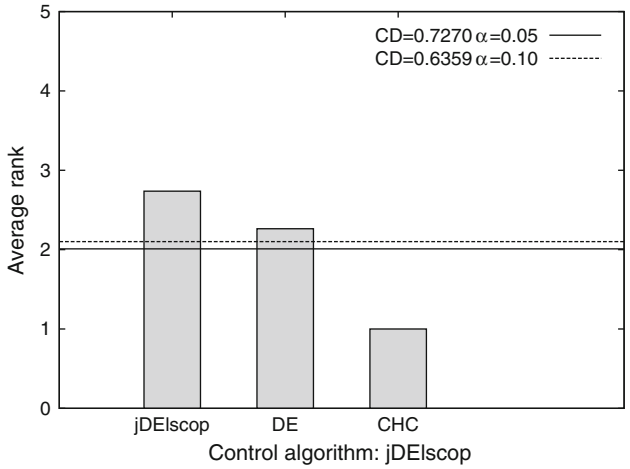

Fig. 8 Bonferroni–Dunn's graphic corresponding to the results when $D = 1{,}000$

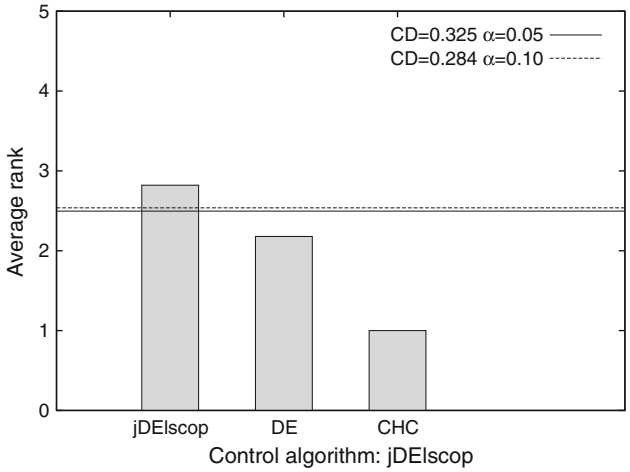

Fig. 9 Bonferroni–Dunn's graphic corresponding to the results on a set of all dimensions

Bonferroni–Dunn's graphics are depicted in Figs. 4, 5, 6 ,7, and 8 for all problem dimensions (denoted by D). On each figure a horizontal cut line is drawn for two levels of significance, $\alpha = 0.05$ and $\alpha = 0.10$. The application of Bonferroni–Dunn's test informs us that jDElscop is significantly better than CHC and G-CMA-ES.

Bonferroni–Dunn's graphic is depicted in Fig. 9 on a set of all dimensions for the jDElscop, CHC, and DE algorithms (without the G-CMA-ES algorithm). The application of Bonferroni–Dunn's test informs us that jDElscop is significantly better than CHC and DE.

5.4.2 *Comparisons among all algorithms*

Post-hoc tests were extended by the performing of other tests: Nemenyi's, Holm's, Shaffer's, and Bergmann–Hommel's. They are based on computation of the adjusted p values. Pairwise comparisons of all algorithms were done between each other.

Tables 22, 23 ,24 ,25, and 26 report adjusted p values, which take into account that multiple tests were conducted. We applied the following methods (García and Herrera 2008; García et al. 2010) to acquire adjusted p values: Nemenyi's test, Holm's procedure, Shaffer's static procedure, and Bergmann–Hommel's dynamic procedure. The mark ✓ indicates the case when a particular test rejects hypothesis for $\alpha = 0.05$.

The adjusted p values for $D = 50$ are presented in Table 22. For $\alpha = 0.10$ and for $\alpha = 0.05$ Nemenyi's, Holm's, Shaffer's reject hypotheses 1–3 and Bergmann–Hommel's procedures rejects hypotheses 1–4.

Table 23 shows the adjusted p values for dimension $D = 100$. The Nemenyi's, Holm's, Shaffer's, and procedures reject hypotheses 1–3 for $\alpha = 0.10$ and for $\alpha = 0.05$. Bergmann–Hommel's procedures rejects hypotheses 1–4 for $\alpha = 0.10$ and for $\alpha = 0.05$.

The adjusted p values for $D = 200$ are presented in Table 24. For $\alpha = 0.10$ and also for $\alpha = 0.05$ Nemenyi's, Holm's, Shaffer's, and Bergmann–Hommel's procedure reject hypotheses 1–3, and for $\alpha = 0.10$ Bergmann–Hommel's procedure rejects hypothesis 4, too.

Table 25 shows the adjusted p values for dimension $D = 500$. For $\alpha = 0.05$ Nemenyi's procedure rejects hypotheses 1–3, while Holm's, Shaffer's, and Bergmann–Hommel's procedures reject hypotheses 1–4. The Nemenyi's, Holm's, Shaffer's, and Bergmann–Hommel's procedures reject hypotheses 1–4 for $\alpha = 0.10$. And for $\alpha = 0.10$ Nemenyi's procedure rejects hypothesis 4, too.

The adjusted p values for $D = 1{,}000$ of three algorithms only (without the G-CMA-ES algorithm) are presented in Table 26. For $\alpha = 0.10$ and $\alpha = 0.05$ Nemenyi's, Holm's,

Table 22 Adjusted p values when $D = 50$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	jDElscop vs. CHC	4.601914E−8	2.761148E−7 ✓	2.761148E−7 ✓	2.761148E−7 ✓	2.761148E−7 ✓
2	jDElscop vs. G-CMA-ES	1.268407E−4	7.610445E−4 ✓	6.342037E−4 ✓	3.805222E−4 ✓	3.805222E−4 ✓
3	DE vs. CHC	2.098472E−4	0.00125908 ✓	8.393891E−4 ✓	6.295418E−4 ✓	6.295418E−4 ✓
4	DE vs. G-CMA-ES	0.03814187	0.22885123	0.11442561	0.11442561	0.03814187 ✓
5	jDElscop vs. DE	0.07854585	0.47127510	0.15709170	0.15709170	0.15709170
6	CHC vs. G-CMA-ES	0.10235752	0.61414515	0.15709170	0.15709170	0.15709170

Table 23 Adjusted p values when $D = 100$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	jDElscop vs. CHC	4.601914E−8	2.761148E−7 ✓	2.761148E−7 ✓	2.761148E−7 ✓	2.761148E−7 ✓
2	jDElscop vs. G-CMA-ES	1.268407E−4	7.610445E−4 ✓	6.342037E−4 ✓	3.805222E−4 ✓	3.805222E−4 ✓
3	DE vs. CHC	2.098472E−4	0.00125908 ✓	8.393891E−4 ✓	6.295418E−4 ✓	6.295418E−4 ✓
4	DE vs. G-CMA-ES	0.03814187	0.22885123	0.11442561	0.11442561	0.03814187 ✓
5	jDElscop vs. DE	0.078545850	0.47127510	0.15709170	0.15709170	0.15709170
6	CHC vs. G-CMA-ES	0.10235752	0.61414515	0.15709170	0.15709170	0.15709170

Table 24 Adjusted p values when $D = 200$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	jDElscop vs. CHC	6.545677E−8	3.927406E−7 ✓	3.927406E−7 ✓	3.927406E−7 ✓	3.927406E−7 ✓
2	DE vs. CHC	2.684031E−4	0.00161041 ✓	0.00134201 ✓	8.052094E−4 ✓	8.052094E−4 ✓
3	jDElscop vs. G-CMA-ES	2.684031E−4	0.00161041 ✓	0.00134201 ✓	8.052094E−4 ✓	8.052094E−4 ✓
4	DE vs. G-CMA-ES	0.05945109	0.35670655	0.17835327	0.17835327	0.05945109
5	CHC vs. G-CMA-ES	0.07854585	0.47127510	0.17835327	0.17835327	0.15709170
6	jDElscop vs. DE	0.07854585	0.47127510	0.17835327	0.17835327	0.15709170

Table 25 Adjusted p values when $D = 500$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	jDElscop vs. CHC	3.598143E−7	2.158885E−6 ✓	2.158885E−6 ✓	2.158885E−6 ✓	2.158885E−6 ✓
2	jDElscop vs. G-CMA-ES	2.559570E−5	1.535742E−4 ✓	1.279785E−4 ✓	7.678712E−5 ✓	7.678712E−5 ✓
3	DE vs. CHC	8.688074E−4	0.00521284 ✓	0.00347522 ✓	0.00260642 ✓	0.00260642 ✓
4	DE vs. G-CMA-ES	0.01427390	0.08564344	0.04282172 ✓	0.04282172 ✓	0.01427390 ✓
5	jDElscop vs. DE	0.07854585	0.47127510	0.15709170	0.15709170	0.15709170
6	CHC vs. G-CMA-ES	0.37907971	2.27447831	0.37907971	0.37907971	0.37907971

Table 26 Adjusted p values when $D = 1{,}000$

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	jDElscop vs. CHC	8.636119E−8	2.590835E−7 ✓	2.590835E−7 ✓	2.590835E−7 ✓	2.590835E−7 ✓
2	DE vs. CHC	9.888398E−5	2.966519E−4 ✓	1.977679E−4 ✓	9.888398E−5 ✓	9.888398E−5 ✓
3	jDElscop vs. DE	0.14429205	0.43287616	0.14429205	0.14429205	0.14429205

Table 27 Adjusted p values on a set of all dimensions

i	Hypothesis	Unadjusted p	p_{Neme}	p_{Holm}	p_{Shaf}	p_{Berg}
1	jDElscop vs. CHC	3.938167E−36	1.181450E−35 ✓	1.181450E−35 ✓	1.181450E−35 ✓	1.181450E−35 ✓
2	DE vs. CHC	4.461321E−16	1.338396E−15 ✓	8.922643E−16 ✓	4.461321E−16 ✓	4.461321E−16 ✓
3	jDElscop vs. DE	9.626145E−6	2.887843E−5 ✓	9.626145E−6 ✓	9.626145E−6 ✓	9.626145E−6 ✓

Shaffer's, and Bergmann–Hommel's procedures reject hypotheses 1–2.

Finally, statistical analysis of the average error was conducted. These well-known statistical tests show that the proposed jDElscop algorithm performs significantly better than the CHC and DE algorithms and performs better or comparable (not significantly better) to the original DE algorithm using the *DE/rand/1/exp* strategy.

The adjusted p values on a set of all dimensions are presented in Table 27 for the jDElscop, CHC, and G-CMA-ES algorithms (without the G-CMA-ES algorithm). For $\alpha = 0.10$ and $\alpha = 0.05$ Nemenyi's, Holm's, Shaffer's, and Bergmann–Hommel's procedures reject hypotheses 1–3, e.g., all hypotheses are rejected.

The statistical procedures in this analysis were based on mean values. Apart from these tests, a comparison through the use of the median measure would also be informative. It might provide another aspect of differences between the algorithms.

6 Conclusions

This paper proposed a jDElscop algorithm. This algorithm was evaluated on a set of benchmark functions provided for the Special Issue on Scalability of Evolutionary Algorithms and other Metaheuristics for Large Scale Continuous Optimization Problems.

The algorithm uses three DE's strategies, a newly proposed population size reduction mechanism, and a mechanism for changing the sign of F control parameter. The self-adaptive control mechanism for changing the control parameters (F and CR) is applied on all three DE strategies.

The obtained results and statistical analysis give evidence that the jDElscop algorithm is a highly competitive algorithm for large scale continuous optimization problems. To summarize the results of the tests, the jDElscop algorithm presents significantly better results than the remaining algorithms, in most cases.

Acknowledgments The authors would like to thank the organizers of this special issue and the reviews for their valuable comments.

References

Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. In: Proceedings of the 2005 IEEE Congress on evolutionary computation, IEEE Press, pp 1769–1776

van den Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimisation. IEEE Trans Evol Comput 3:225–239

Brest J (2009) Constrained real-parameter pptimization with ε-self-adaptive differential evolution. Stud Comput Intell. ISBN: 978-3-642-00618-0 198:73–93

Brest J, Maučec MS (2008) Population size reduction for the differential evolution algorithm. Appl Intell 29(3):228–247

Brest J, Greiner S, Bošković B, Mernik M, Žumer V (2006a) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657

Brest J, Žumer V, Maučec MS (2006b) Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: The 2006 IEEE Congress on evolutionary computation CEC 2006, IEEE Press, pp 919–926

Brest J, Bošković B, Greiner S, Žumer V, Maučec MS (2007) Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Comput Fusion Found Methodol Appl 11(7):617–629

Brest J, Zamuda A, Bošković B, Maučec MS, Žumer V (2008) High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In: 2008 IEEE World Congress on computational intelligence, IEEE Press, pp 2032–2039

Brest J, Zamuda A, Bošković B, Maučec MS, Žumer V (2009) Dynamic optimization using self-adaptive differential evolution. In: IEEE Congress on evolutionary computation (CEC) 2009, IEEE Press, pp 415–422

Brest J, Zamuda A, Bošković B, Fister I, Maučec MS (2010) Large scale global optimization using self-adaptive differential evolution algorithm. In: IEEE World Congress on computational intelligence, pp 3097–3104

Caponio A, Neri F, Tirronen V (2009) Super-fit control adaptation in memetic differential evolution frameworks. Soft Comput Fusion Found Methodol Appl 13(8):811–831

Das S, Abraham A, Chakraborty U, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evol Comput 13(3):526–553

Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30

Dunn OJ (1961) Multiple comparisons among means. J Am Stat Assoc 56(293):52–64

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemate. In: Whitley LD (ed) Foundations of genetic algorithms, vol 2. Morgan Kaufmann Publishers, USA, pp 187–202

Feoktistov V (2006) Differential evolution: in search of solutions (Springer optimization and Its applications). Springer, New York

Gao Y, Wang YJ (2007) A memetic differential evolutionary algorithm for high dimensional functions' optimization. Int Conf Nat Comput 4:188–192. doi:10.1109/ICNC.2007.60

García S, Herrera F (2008) An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. J Mach Learn Res 9:2677–2694

García S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC 2005 Special Session on Real Parameter Optimization. Journal of Heuristic 15(6):617–644

García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Info Sci 180(10):2044–2064

Liu Y, Yao X, Zhao Q, Higuchi T (2001) Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of the 2001 Congress on evolutionary computation CEC 2001, IEEE Press, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, pp 1101–1108. http://citeseer.ist.psu.edu/liu01scaling.html

MacNish C (2007) Towards unbiased benchmarking of evolutionary and hybrid algorithms for real-valued optimisation. Connect Sci 19(4):225–239

Mladenovic N, Drazic M, Kovacevic-Vujcic V, Cangalovic M (2008) General variable neighborhood search for the continuous optimization. Eur J Oper Res 191(3):753–770

Molina D, Lozano M, Herrera F (2009a) Memetic algorithm with local search chaining for continuous optimization problems: a scalability test. In: Proceedings of the ninth international conference intelligent systems design and applications computation, IEEE Press, pp 1068–1073

Molina D, Lozano M, Herrera F (2009b) Memetic algorithm with local search chaining for large scale continuous optimization problems. In: Proceedings of the 2009 IEEE Congress on evolutionary computation, CEC '09, pp 830–837

Muelas S, LaTorre A, Penã JM (2009) A memetic differential evolution algorithm for continuous optimization. In: Proceedings of the ninth international conference intelligent systems design and applications computation, IEEE Press, pp 1080–1084

Neri F, Tirronen V (2010) Recent advances in differential evolution: a survey and experimental analysis. Artif Intell Rev 33(1–2): 61–106

Potter MA, De Jong K (1994) A cooperative coevolutionary approach to function optimization. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel problem solving from nature, PPSN III, Springer, Berlin, pp 249–257. http://citeseer.ist.psu.edu/potter94cooperative.html

Price KV, Storn RM, Lampinen JA (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin

Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE Congress on evolutionary computation CEC 2005, vol 2, IEEE Press, pp 1785–1791

Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13(2):398–417

Rahnamayan S, Tizhoosh H, Salama M (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79

Rönkkönen J, Kukkonen S, Price KV (2005) Real-parameter optimization with differential evolution. In: The 2005 IEEE Congress on evolutionary computation CEC 2005, vol 1, IEEE Press, pp 506–513

Sofge D, De Jong K, Schultz A (2002) A blended population approach to cooperative coevolution for decomposition of complex problems. In: Proceedings of the 2002 Congress on evolutionary computation (CEC 2002), IEEE, pp 413–418

Storn R, Price K (1995) Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-012, Berkeley, CA. http://citeseer.ist.psu.edu/article/storn95differential.html

Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359

Tang K, Yao X, Suganthan PN, MacNish C, Chen YP, Chen CM, Yang Z (2007) Benchmark functions for the CEC 2008 Special Session and Competition on High-Dimenasional Real-Parameter Optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China. http://nical.ustc.edu.cn/cec08ss.php

Tang K, Li X, Suganthan PN, Yang Z, Weise T (2009) Benchmark functions for the CEC 2010 Special Session and Competition on Large Scale Global Optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China

Teng N, Teo J, Hijazi M (2009) Self-adaptive population sizing for a tune-free differential evolution. Soft Comput Fusion Found Methodol Appl 13(7):709–724

Teo J (2006) Exploring dynamic self-adaptive populations in differential evolution. Soft Comput Fusion Found Methodol Appl 10(8):673–686

Wang Y, Li B (2008) A restart univariate estimation of distribution algorithm: sampling under mixed Gaussian and Lévy probability distribution. In: 2008 IEEE World Congress on computational intelligence, IEEE Press, pp 3917–3924

Yang Z, Tang K, Yao X (2007) Differential evolution for high-dimensional function optimization. In: Srinivasan D, Wang L (eds) 2007 IEEE Congress on evolutionary computation, IEEE computational intelligence society, IEEE Press, pp 3523–3530

Yang Z, Tang K, Yao X (2008a) Large scale evolutionary optimization using cooperative coevolution. Info Sci 178(15):2985–2999

Yang Z, Tang K, Yao X (2008b) Multilevel cooperative coevolution for large scale optimization. In: Proc. IEEE World Congress on computational intelligence (WCCI 2008), IEEE Press, pp 1663–1670

Yang Z, Tang K, Yao X (2008c) Self-adaptive differential evolution with neighborhood search. In: 2008 IEEE World Congress on computational intelligence evolutionary computation, IEEE Press, pp 1110–1116

Zaharie D (2002) Critical values for the control parameters of differential evolution algorithms. In: Proc. of Mendel 2002, 8th international conference on soft computing, pp 62–67

Zamuda A, Brest J, Bošković B, Žumer V (2008) Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In: 2008 IEEE World Congress on computational intelligence, IEEE Press, pp 3719–3726

Zar JH (1999) Biostatistical analysis. Prentice-Hall, Englewood Cliffs

Zhang J, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958

Zhao SZ, Liang JJ, Suganthan PN, Tasgetiren MF (2008) Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: 2008 IEEE World Congress on computational intelligence, IEEE Press, pp 3845–3852