Self-Adaptive Differential Evolution Algorithm with a Small and Varying Population Size

Janez Brest, *Member, IEEE*, Borko Bošković, Aleš Zamuda, *Student Member, IEEE*,
Iztok Fister, *Member, IEEE*, Mirjam Sepesy Maučec, *Member, IEEE*

***Abstract*— In this paper we present self-adaptive differential evolution algorithm with a small and varying population size on large scale global optimization. The experimental results obtained by our algorithm on benchmark functions provided for the CEC 2012 competition and special session on Large Scale Global Optimization are presented. The experiments were performed on 20 test functions with high dimension $D = 1000$. Obtained results show that our algorithm performs highly competitive in comparison with the algorithms presented at similar CEC 2010 competition.**

I. Introduction

IN this paper Large Scale Global Optimization (LSGO) is addressed. Recently, increasing research efforts on handling LSGO have been reported, and a variety of algorithms have been presented. In LSGO, one important task is to developed a proper algorithm for optimization of problems with high dimensionality.

The global optimization problem can be defined as follows. We need to find variables of vector $\vec{x} = \{x_1, x_2, ..., x_D\}$, where D denotes the dimensionality of a problem, such that objective function $f(\vec{x})$ is optimized (minimized in this paper). Domains of the variables are defined by their lower and upper bounds $x_{j,low}, x_{j,upp}$ for $j = 1, 2, ..., D$. Therefore, such an optimization is called bound-constrained optimization.

Differential Evolution (DE) [14], [7], [12] is a floating-point encoding evolutionary algorithm. In recent decades, it demonstrates its efficiency and robustness when solving real-world optimization problems [7]. This simple yet powerful algorithm becomes very popular as global optimizer, especially over continuous spaces.

DE has three control parameters: F is amplification factor of the difference vector, CR is crossover control parameter, and NP is population size. All three control parameters are fixed during the optimization process in the original DE algorithm, proposed by Storn and Price in 1995. A tuning of the control parameters can be used to find out good values for them before actual optimization process starts. In literature, various adaptation mechanisms are proposed to overcome the tuning problems of DE parameters [13], [1], [11], [20], [16].

In this paper a performance evaluation of our self-adaptive jDEsps algorithm is evaluated on the benchmark functions provided for the CEC 2012[1] special session on LSGO [15]. LSGO as an active research area was presented also at CEC 2008 and CEC 2010.

J. Brest, B. Bošković, A. Zamuda, I. Fister, and M. Sepesy Maučec are with the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia, (email: janez.brest@uni-mb.si).

[1]CEC 2012 has the same set of benchmark functions as CEC 2010.

The structure of the paper is as follows. An overview of related work is presented in Section II. Section III gives background for this work, where an overview of DE, description of the self-adaptive control parameters, population size reduction mechanisms, and outlines of our two previous algorithms are given. Section IV presents a description of our new variant algorithm, called jDEsps , which is used in these experiments. In Section V experimental results of the jDEsps algorithm on benchmark functions are presented. Section VI concludes the paper with some final remarks.

II. Related Work

The DE [14] algorithm has become a very powerful tool for global continuous optimization problems [19], which have been one of the most interesting trends in the last decades, because they appear in many real-world problems [10]. The original DE that was proposed first in 1995, keeps all three control parameters fixed during optimization process. In order to improve algorithm's performance, many new versions of the DE were proposed later in various research areas [12], [7], [20], [6]. In literature, a lot of improved versions of DE can be found that are dealing with the control parameters F and CR. Adaptive and/or self-adaptive techniques were introduced [1], [13] among many others. On the other hand, only a few researches are related with the third DE control parameter, i.e., population size, NP [16], [2], [4]. Some researches outlined that NP plays an important role among control parameters, and even more that NP is the most important DE parameter, i.e., a performance of the DE algorithm is determined by setting appropriate value of the population size.

Recently, we proposed a self-adaptive jDE algorithm [1]. The jDE-based variants were used for unconstrained optimization, dynamic problems, constrained optimization, some variants also for multi-objective optimization, etc. The jDE-based algorithms were applied to solve the following LSGO problems: CEC 2008 [5], CEC 2010 [4], and large-scale continuous optimization problems [3].

In Special Issue on Scalability of Evolutionary Algorithms and Other Metaheuristics for Large-Scale Continuous Optimization Problems seven DE-based algorithms [18], [8], [17], [19], [3], [22], [9] among thirteen were proposed, which might also reflects usefulness of the DE algorithm for LSGO problems.

Recently, the SaDE-MMTS, which includes SaDE enhanced with JADE mutation and modified multi-trajectory search (MMTS) is proposed in [21] for solving LSGO problems.

1: {*pop* ... population}
2: {$imin$ ... index of currently best individual}
3: {$\vec{x}_i$ ... i-th individual of population}
4: {*MAX_FEs* ... maximum number of function evaluations}
5: Initialization() {Generate uniformly distributed random population}
6: **for** $(it = 0;\ it < MAX\_FEs;\ it++)$ **do**
7: $i = it$ **mod** *NP* {mod ... modulo operation}
8: {Try to improve currently best individual two times per generation. Perform these steps only for two selected individuals (when $i = 0$ or $i = 1$), and when iteration exceeded half of MAX_FEs.}
9: **if** $((it > MAX\_FEs/2)$ **and** $(i < 2))$ **then**
10: $\vec{x}_i = InitRand()$ {Initialize randomly individual $\vec{x}_i$ (without function evaluation)}
11: {*** Perform one iteration of jDE using strategy to improve currently best individual***}
12: $F_l = 0.01$ {Lower bound of F control parameter}
13: $F_u = 0.09$ {Upper bound of F control parameter}
14: $\vec{u}$ =jDEoneIter($imin$, *pop*) {Perform one iteration of the jDE.}
15: **else**
16: **if** i **mod** 2 == 0 **then**
17: {Otherwise perform one iteration of the jDE with bin crossover using values $F_l = 0.1; F_u = 0.9$; on individual $\vec{x}_i$}
18: $\vec{u}$ =jDEoneIter(i, *pop*)
19: **else**
20: {Otherwise perform one iteration of the jDE with exp crossover using values $F_l = 0.4; F_u = 0.6$; and CR from interval $[0.6, 1.0]$ on individual $\vec{x}_i$}
21: $\vec{u}$ =jDEoneIter(i, *pop*)
22: **end if**
23: **end if**
24: $f(\vec{u})$ {Function evaluation}
25: ... {Selection, etc.}
26: **end for**

Algorithm 1: jDElsgo [5] algorithm

1: {*pop* ... population}
2: {$imin$ ... index of currently best individual}
3: {$\vec{x}_i$ ... i-th individual of population}
4: {*MAX_FEs* ... maximum number of function evaluations}
5: {s ... one of strategies ($jDEbin$, $jDEexp$, $jDEbest$)}
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 $\vec{x}_i^{(G)}$}
19: {The jDE applies mutation and crossover to generate trial vector $\vec{u}^{(G)}$}
20: $\vec{u}^{(G)}$ =jDE(i, *pop*, s)
21: {Fitness evaluation and selection}
22: **if** $(f(\vec{u}^{(G)}) \leq f(\vec{x}_i^{(G)}))$ **then**
23: $\vec{x}_i^{(G+1)} = \vec{u}^{(G)}$
24: **end if**
25: **end for**

Algorithm 2: jDElscop [3] algorithm

III. Background

In this section, we give some backgrounds that are needed in Section IV.

A. The Differential Evolution

DE is a population-based algorithm. It creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness value.

The population of the original DE algorithm [14] consists of NP vectors:

$$\vec{x}_i^{(G)} = (x_{i,1}^{(G)}, x_{i,2}^{(G)}, ..., x_{i,D}^{(G)}),\ i = 1, 2, ..., NP,$$

where G denotes the generation. For each D-dimensional vector during one generation, DE employs the mutation and crossover operations to produce a trial vector $\vec{u}_i^{(G)}$. Then selection operation is used to choose vectors for the next generation $(G + 1)$.

A mutant vector $\vec{v}_i^{(G)}$ is created by using one of the DE mutation strategies. The mostly used is 'rand/1' strategy, which can be described as follows:

$$\vec{v}_i^{(G)} = \vec{x}_{r_1}^{(G)} + F \cdot (\vec{x}_{r_2}^{(G)} - \vec{x}_{r_3}^{(G)}).$$

Indexes r_1, r_2, and r_3 denote random integers within set $\{1, NP\}$ and $r_1 \neq r_2 \neq r_3 \neq i$.

After mutation, a crossover operation forms the trial vector $\vec{u}_i^{(G)}$:

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

for $i = 1, 2, ..., NP$ and $j = 1, 2, ..., D$. Crossover parameter CR is within the range $[0, 1)$ and presents the probability of creating components for trial vector from a mutant vector. Index $j_{rand} \in \{1, NP\}$ is a randomly chosen integer that is responsible for the trial vector containing at least one component from the mutant vector.

The selection operation for a minimization problem is defined as follows:

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

B. Self-Adaptation of F and CR

The jDE algorithm [1] uses self-adapting mechanism of two control parameters. Each individual is extended with control parameter values F and CR as follows:

$$\vec{x}_i^{(G)} = (x_{i,1}^{(G)}, x_{i,2}^{(G)}, ..., x_{i,D}^{(G)}, F_i^{(G)}, CR_i^{(G)}).$$

New control parameters $F_i^{(G+1)}$ and $CR_i^{(G+1)}$ are calculated before the mutation is performed as follows [1]:

$$F_i^{(G+1)} = \begin{cases} F_l + rand_1 * F_u, & \text{if } rand_2 < \tau_1, \\ F_i^{(G)}, & \text{otherwise,} \end{cases}$$

$$CR_i^{(G+1)} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2, \\ CR_i^{(G)}, & \text{otherwise,} \end{cases}$$

where $rand_j$, for $j \in \{1, 2, 3, 4\}$ are uniform random values within the range $[0, 1]$.

In [1] and in our later works parameters τ_1, τ_2, F_l, F_u are fixed to values $0.1, 0.1, 0.1, 0.9$, respectively. The new F takes a value from $[0.1, 1.0]$, and the new CR from $[0, 1]$.

C. Population Size Reduction

The third DE control parameter NP is usually fixed during the optimization process. Recently, some works are related to changing NP during optimization process, but there are still no general guidelines how to set NP at the beginning of optimization process, when to change (question if to increase or decrease remains open, too), etc. Since all parameters are related to each other and they are depending on optimization problem that are solving, further works are necessary.

Recently, a population size reduction mechanism was proposed in [2]. In the reduction scheme, the new population size is equal to half the last population size. This mechanism was used in [5], [4], [3].

Let $pmax$ be the number of different population sizes used in the evolutionary process. Then $pmax - 1$ reductions need to be performed.

D. The jDElsgo and jDElscop Algorithms

For solving high-dimensional real-parameter optimization problems on CEC 2008 we used *jDEdynNP-F* algorithm [5], while for solving LSGO problems on CEC 2010 we implemented jDElsgo algorithm [4]. The pseudo-code of the jDElsgo algorithm is presented at Algorithm 1.

Algorithm 2 presents a pseudo-code of jDElscop algorithm [3] which was applied to solve large-scale continuous optimization problems [10].

IV. The proposed Algorithm

In this section our new algorithm jDEsps for solving large-scale global optimization is presented. The jDEsps algorithm is an extended version of the jDElsgo [4] and jDElscop [3] algorithms.

The new proposed algorithm uses:

- more DE strategies,
- self-adaptive F and CR control parameters,
- population size reduction mechanism,
- control parameter F sign change mechanism, and
- ageing.

The pseudo-code of new jDEsps algorithm is presented in Algorithms 3 and 4. The main features that are proposed in the jDEsps algorithm are:

- Small and varying population size at the beginning of the optimization process. Later, the algorithm increases population size and after then it applies population size reduction mechanism.
- 'Local search' is applied on the best individual using $jDELS$ strategy.

1: {*pop* ... population}
2: {*NP* ... population size}
3: {$imin$ ... index of currently best individual}
4: {$\vec{x}_i$ ... i-th individual of population}
5: {$MaxFEs$ ... maximum number of function evaluations}
6: {$rand(0, 1)$ ... uniformly distributed random number $[0, 1)$}
7: Initialization()
8: *NP* = 100 {max. population size}
9: {** Generate uniformly distributed random population within search space **}
10: **for** ($it = 0$; $it < MaxFEs$; $it = it + 1$) **do**
11: $i = it$ **mod** *NP* {mod ... modulo operation}
12:
13: *NP* = selectPopSize() {** a part of population is used **}
14: $F_l = 0.1 + \sqrt{\frac{1}{NP}}$; $F_u = 1.0$;
15: {** Perform one iteration of the jDE using one of four strategies. EXP crossover is used for jDEexp, otherwise BIN crossover is used. **}
16:
17: **if** ($rand(0, 1) < 0.1$) **then**
18: {** jDELS is applied on the best individual. ** }
19: *CR* $\in [0.0, 1.0]$
20: $\vec{u}$ =jDELS($imin$, *pop*)
21: **else**
22: **if** ($rand(0, 1) < 0.1$ **and** $it > \frac{MAX\_FES1}{2}$) **then**
23: {** jDEw **}
24: $F_l = 0.4$; $F_u = 1.0$; *CR* $\in [0.6, 1.0]$
25: $\vec{u}$ =jDEw(i, *pop*)
26: **else**
27: **if** ($i < \frac{NP}{2}$) **then**
28: {** jDEbin **}
29: $F_l = 0.1 + \sqrt{\frac{1}{NP}}$; $F_u = 1.0$; *CR* $\in [0.0, 0.95]$
30: $\vec{u}$ =jDEbin(i, *pop*)
31: **else**
32: {** jDEexp **}
33: $F_l = 0.1 + \sqrt{\frac{1}{NP}}$; $F_u = 1.0$; *CR* $\in [0.5, 0.95]$
34: $\vec{u}$ =jDEexp(i, *pop*)
35: **end if**
36: **end if**
37: **end if**
38: {Bound check; if volatile then set on bound}
39: $f(\vec{u})$ {Function evaluation}
40: ... {Selection, etc.}
41: {Population size reduction}
42: **end for**

Algorithm 3: jDEsps algorithm

- A new strategy, called $jDEw$, is used. This strategy tries to move forward the best individual using large step movement,
- Two already known $jDEexp$ and $jDEbin$ strategies both with F sign change mechanism.

The $jDELS$ strategy is:

$$smallF = \frac{F}{1.0 + 10 * rand(0, 1)},$$

$$\vec{v}_{best}^{(G)} = \vec{x}_{best}^{(G)} + smallF \cdot (\vec{x}_{r_2}^{(G)} - \vec{x}_{r_3}^{(G)}).$$

A value $smallF$, using self-adaptive F value, is computed and then applied in the mutation operation on the best individuals. The $jDELS$ strategy performs 'bin' crossover. This strategy can be interpreted as a local search and it is

TABLE I

PROPERTIES OF THE CEC 2010 BENCHMARK FUNCTIONS [15]

Function	Modality	Shifted	Separability	Scalability	x_i
F_1 : Shifted Elliptic Function	Unimodal	Shifted	Separable	Scalable	[−100, 100]
F_2 : Shifted Rastrigin's Function	Multimodal	Shifted	Separable	Scalable	[−5, 5]
F_3 : Shifted Ackley's Function	Multimodal	Shifted	Separable	Scalable	[−32, 32]
F_4 : Single-group Shifted and m-rotated Elliptic Function	Unimodal	Shifted	Single-group m-rotated	Single-group m-nonseparable	[−100, 100]
F_5 : Single-group Shifted and m-rotated Rastrigin's Function	Multimodal	Shifted	Single-group m-rotated	Single-group m-nonseparable	[−5, 5]
F_6 : Single-group Shifted and m-rotated Ackley's Function	Multimodal	Shifted	Single-group m-rotated	Single-group m-nonseparable	[−32, 32]
F_7 : Single-group Shifted m-dimensional Schwefel's Function	Unimodal	Shifted	-	Single-group m-nonseparable	[−100, 100]
F_8 : Single-group Shifted m-dimensional Rosenbrock's Function	Multimodal	Shifted	-	Single-group m-nonseparable	[−100, 100]
F_9 : $\frac{D}{2m}$-group Shifted and m-rotated Elliptic Function	Unimodal	Shifted	$\frac{D}{2m}$-group m-rotated	$\frac{D}{2m}$-group m-nonseparable	[−100, 100]
F_{10} : $\frac{D}{2m}$-group Shifted and m-rotated Rastrigin's Function	Multimodal	Shifted	$\frac{D}{2m}$-group m-rotated	$\frac{D}{2m}$-group m-nonseparable	[−5, 5]
F_{11} : $\frac{D}{2m}$-group Shifted and m-rotated Ackley's Function	Multimodal	Shifted	$\frac{D}{2m}$-group m-rotated	$\frac{D}{2m}$-group m-nonseparable	[−32, 32]
F_{12} : $\frac{D}{2m}$-group Shifted m-dimensional Schwefel's Problem 1.2	Unimodal	Shifted	-	$\frac{D}{2m}$-group m-nonseparable	[−100, 100]
F_{13} : $\frac{D}{2m}$-group Shifted m-dimensional Rosenbrock's Function	Multimodal	Shifted	-	$\frac{D}{2m}$-group m-nonseparable	[−100, 100]
F_{14} : $\frac{D}{m}$-group Shifted and m-rotated Elliptic Function	Unimodal	Shifted	$\frac{D}{m}$-group m-rotated	$\frac{D}{m}$-group m-nonseparable	[−100, 100]
F_{15} : $\frac{D}{m}$-group Shifted and m-rotated Rastrigin's Function	Multimodal	Shifted	$\frac{D}{m}$-group m-rotated	$\frac{D}{m}$-group m-nonseparable	[−5, 5]
F_{16} : $\frac{D}{m}$-group Shifted and m-rotated Ackley's Function	Multimodal	Shifted	$\frac{D}{m}$-group m-rotated	$\frac{D}{m}$-group m-nonseparable	[−32, 32]
F_{17} : $\frac{D}{m}$-group Shifted m-dimensional Schwefel's Problem 1.2	Unimodal	Shifted	-	$\frac{D}{m}$-group m-nonseparable	[−100, 100]
F_{18} : $\frac{D}{m}$-group Shifted m-dimensional Rosenbrock's Function	Multimodal	Shifted	-	$\frac{D}{m}$-group m-nonseparable	[−100, 100]
F_{19} : Shifted Schwefel's Problem 1.2	Unimodal	Shifted	-	Fully-nonseparable	[−100, 100]
F_{20} : Shifted Rosenbrock's Function	Multimodal	Shifted	-	Fully-nonseparable	[−100, 100]

```
{NP ... population size}
{it ... iteration (0..MaxFEs − 1)}
{Constant values used to determine NP value}
{ MaxFES1 = 1.2e5}
{ MaxFES2 = 6e5}
{ NP_init = 100}
if (it ≥ NP_init and it < MaxFES1/2) then
   NP = 50
else
   if (it < MaxFES1) then
      NP = 10
   else
      if (it < MaxFES2/2) then
         NP = 20
      else
         if (it < MaxFES2) then
            NP = 10
         else
            if (it == MaxFES2) then
               NP = 80
               {hereafter, the pop. size reduction is applied}
            end if
         end if
      end if
   end if
end if
{return NP }
```

Algorithm 4: Population size selection algorithm

highly useful since the number of function evaluations is restricted in this LSGO algorithm.

Another strategy is applied on best individual and it is called $jDEw$. This strategy tries to move forward the best individual using larger step movement as compared to the $jDELS$ strategy. It can be described as follows:

$$F_a = \begin{cases} 2 - F, & \text{if } rand(0,1) \leq 0.5, \\ F, & \text{otherwise,} \end{cases}$$

$$\vec{v}_i^{(G)} = F_a * \vec{x}_{best}^{(G)} + F(\vec{x}_{r_2}^{(G)} - \vec{x}_{r_3}^{(G)}).$$

If an individual did not improved for some generations (we set this value to 300) then it is reinitialized with probability of 0.2.

The main part of the jDEsps algorithm is similar to the jDElscop algorithm as well as the jDElsgo algorithm.

The initial population is selected uniform randomly between the lower $x_{j,low}$ and upper $x_{j,upp}$ bounds defined for each variable x_j.

Let us look at the population size value in our previous algorithms and compared it to the value used in the proposed jDEsps algorithm. The *jDEdynNP-F* algorithm in CEC 2008 used $NP_{init} = D$ and $pmax = 6$ when $D = 1000$, while the jDElsgo algorithm in this research uses $NP_{init} = 400$ and $pmax = 5$.

The proposed jDEsps algorithm uses population size settings as shown in Algorithm 4. Here, we used smaller value for population size.

V. RESULTS

A. Benchmark Functions

The jDEsps algorithm was tested on a test of 20 benchmark functions [15]. The dimension of benchmark functions was $D = 1000$, and 25 runs of algorithm were needed for each function. The optimal values are known for all benchmark functions.

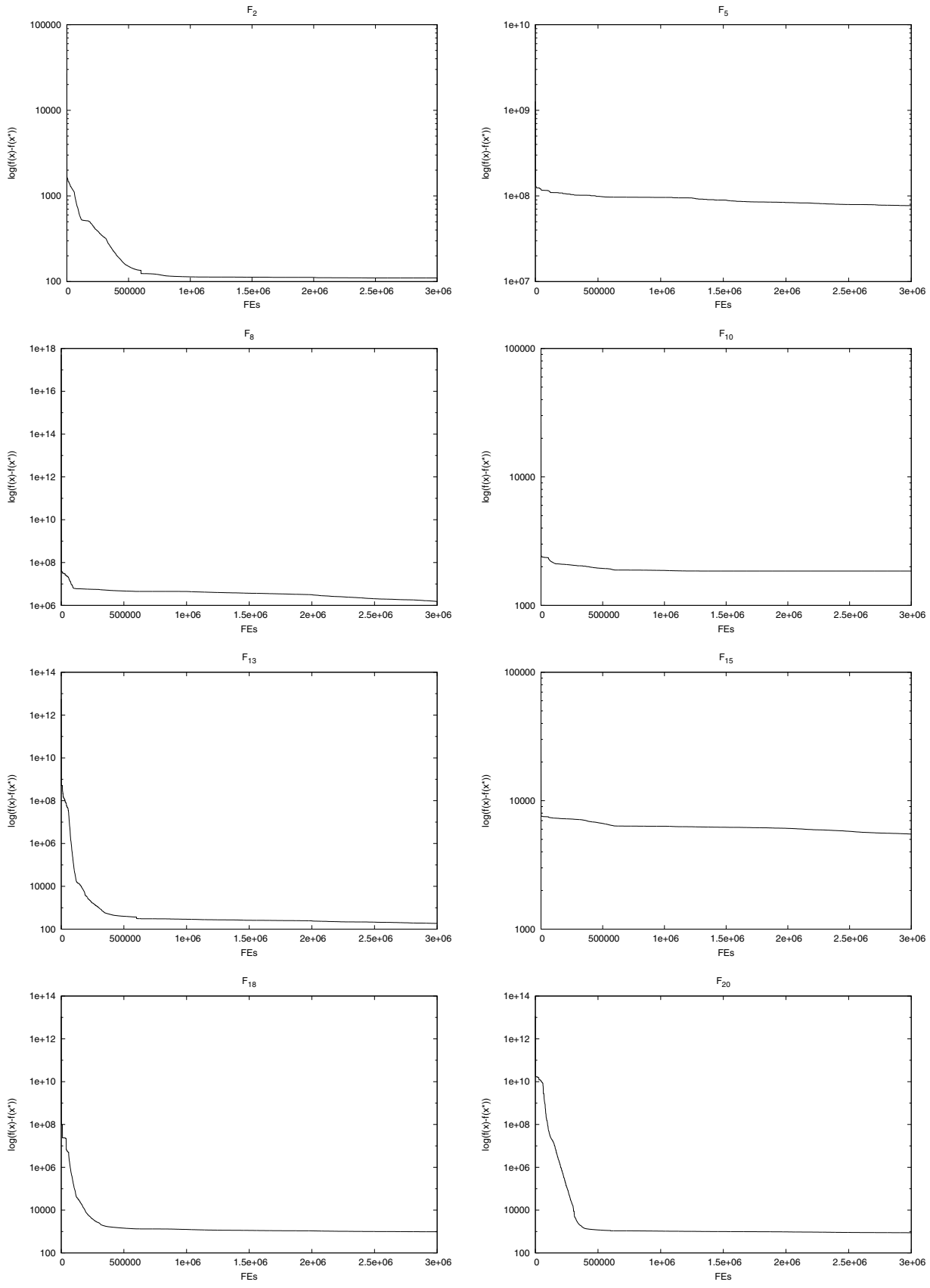

Fig. 1. Convergence Graph for Functions F_2, F_5, F_8, F_{10}, F_{13}, F_{15}, F_{18}, and F_{20}

TABLE II

EXPERIMENTAL RESULTS WITH DIMENSION $D = 1000$.

		F_1	F_2	F_3	F_4	F_5	F_6	F_7
	Best	2.96e-16	2.11e+01	7.86e-14	2.99e+11	6.60e+07	7.55e-09	1.19e+03
	Median	1.87e+02	1.02e+02	1.45e-02	5.98e+11	1.08e+08	8.40e-03	8.45e+04
$1.2e+5$	Worst	4.23e+08	5.42e+03	1.04e+01	7.61e+13	2.06e+08	2.06e+06	1.21e+10
	Mean	2.96e+07	5.25e+02	5.29e-01	4.74e+12	1.10e+08	8.24e+04	5.22e+08
	Std	1.03e+08	1.41e+03	2.11e+00	1.52e+13	3.19e+07	4.12e+05	2.42e+09
	Best	1.60e-23	2.11e+01	7.51e-14	2.93e+11	6.60e+07	7.55e-09	5.71e+02
	Median	3.18e-06	9.59e+01	1.71e-06	5.65e+11	9.51e+07	6.65e-04	4.65e+04
$6.0e+5$	Worst	5.88e+01	4.02e+02	7.46e-04	1.30e+13	1.24e+08	9.35e+00	5.58e+08
	Mean	2.54e+00	1.35e+02	3.94e-05	2.01e+12	9.69e+07	3.75e-01	2.33e+07
	Std	1.17e+01	1.07e+02	1.50e-04	3.46e+12	1.72e+07	1.87e+00	1.11e+08
	Best	2.63e-26	2.01e+01	5.73e-14	2.62e+11	5.01e+07	4.00e-09	3.00e+02
	Median	2.00e-23	8.20e+01	1.03e-13	4.78e+11	7.99e+07	2.18e-08	7.94e+03
$3.0e+6$	Worst	1.82e-22	3.29e+02	4.48e-13	4.71e+12	9.27e+07	1.34e-01	1.36e+07
	Mean	4.10e-23	1.10e+02	1.30e-13	8.15e+11	7.71e+07	5.58e-03	5.77e+05
	Std	4.77e-23	8.29e+01	9.18e-14	9.46e+11	1.09e+07	2.67e-02	2.72e+06

		F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
	Best	1.54e-11	1.19e+04	1.81e+03	2.07e-13	5.28e+02	3.21e-01	4.58e+04
	Median	2.10e+01	4.31e+04	1.84e+03	2.52e-02	4.08e+03	3.31e+02	1.02e+06
$1.2e+5$	Worst	3.90e+07	7.51e+06	8.19e+03	2.26e+02	1.57e+06	2.39e+05	2.15e+09
	Mean	5.99e+06	6.51e+05	2.11e+03	1.73e+01	8.36e+04	1.74e+04	8.90e+07
	Std	1.19e+07	1.73e+06	1.27e+03	5.95e+01	3.12e+05	5.79e+04	4.30e+08
	Best	3.69e-18	1.11e+04	1.81e+03	2.07e-13	4.45e+02	2.09e-01	3.57e+04
	Median	1.82e-03	2.48e+04	1.83e+03	1.07e-05	4.08e+03	2.66e+02	5.52e+05
$6.0e+5$	Worst	2.90e+07	7.37e+05	2.69e+03	1.95e+02	4.49e+05	1.55e+03	3.38e+08
	Mean	4.48e+06	6.74e+04	1.88e+03	1.36e+01	3.26e+04	3.65e+02	1.43e+07
	Std	9.56e+06	1.45e+05	1.73e+02	4.77e+01	9.18e+04	4.05e+02	6.75e+07
	Best	2.34e-19	9.11e+03	1.81e+03	1.58e-13	3.01e+02	2.27e-02	2.55e+04
	Median	2.01e-17	1.63e+04	1.83e+03	2.36e-13	2.02e+03	6.18e+01	3.15e+05
$3.0e+6$	Worst	1.13e+07	1.36e+05	1.97e+03	4.85e-04	1.95e+05	7.32e+02	1.57e+06
	Mean	1.52e+06	2.31e+04	1.85e+03	1.94e-05	1.57e+04	1.86e+02	3.85e+05
	Std	3.63e+06	2.50e+04	4.66e+01	9.70e-05	4.05e+04	2.32e+02	3.54e+05

		F_{15}	$F_{16}*$	F_{17}	F_{18}	F_{19}	F_{20}
	Best	5.85e+03	3.57e+00	4.78e+03	6.62e+02	5.26e+04	2.13e+02
	Median	7.12e+03	3.65e+00	1.71e+04	1.55e+03	1.26e+06	1.94e+03
$1.2e+5$	Worst	9.56e+03	3.98e+02	2.24e+06	8.68e+05	4.25e+06	6.50e+08
	Mean	7.31e+03	2.07e+01	1.72e+05	4.38e+04	1.57e+06	2.60e+07
	Std	9.95e+02	7.88e+01	4.56e+05	1.73e+05	8.90e+05	1.30e+08
	Best	5.54e+03	3.57e+00	3.62e+03	6.46e+02	5.19e+04	1.76e+02
	Median	6.32e+03	3.57e+00	1.46e+04	1.11e+03	1.08e+06	1.17e+03
$6.0e+5$	Worst	7.44e+03	3.03e+02	9.08e+05	3.68e+03	2.57e+06	2.60e+03
	Mean	6.36e+03	1.56e+01	9.48e+04	1.33e+03	1.28e+06	1.11e+03
	Std	5.22e+02	5.99e+01	2.00e+05	6.70e+02	5.92e+05	6.31e+02
	Best	4.82e+03	3.57e+00	2.90e+03	5.67e+02	5.19e+04	1.19e+02
	Median	5.48e+03	3.57e+00	9.95e+03	9.20e+02	8.14e+05	9.70e+02
$3.0e+6$	Worst	6.31e+03	3.84e+01	4.52e+05	1.76e+03	1.37e+06	1.61e+03
	Mean	5.50e+03	4.97e+00	5.52e+04	9.73e+02	8.00e+05	8.79e+02
	Std	3.60e+02	6.97e+00	1.07e+05	2.86e+02	3.34e+05	4.87e+02

Properties of these functions are presented in Table I. Note, although some of used functions are separable in their original form, applying techniques as Salomon's random coordinate rotation technique, make them non-separable [15]. In addition, the global optimum can be shifted. In our research, variables grouping based on separability was not used.

TABLE III

COMPARISON OF EXPERIMENTAL RESULTS WHEN $FEs = 3.0e+6$

FEs		F_1	F_2	F_3	F_4	F_5	F_6	F_7
DECC-G	Best	1.63e-07	1.25e+03	1.20e+00	7.78e+12	1.50e+08	3.89e+06	4.26e+07
	Median	2.86e-07	1.31e+03	1.39e+00	1.51e+13	2.38e+08	4.80e+06	1.07e+08
	Worst	4.84e-07	1.40e+03	1.68e+00	2.65e+13	4.12e+08	7.73e+06	6.23e+08
	Mean	2.93e-07	1.31e+03	1.39e+00	1.70e+13	2.63e+08	4.96e+06	1.63e+08
	Std	8.62e-08	3.26e+01	9.73e-02	5.37e+12	8.44e+07	8.02e+05	1.37e+08
DECC-G*	Best	6.33e-12	4.21e+02	2.23e-08	9.76e+11	2.08e+08	5.07e-03	3.45e+06
	Median	8.97e-12	4.43e+02	3.30e-08	1.96e+12	2.49e+08	8.85e-03	1.04e+07
	Worst	1.31e-11	4.57e+02	4.16e-08	5.39e+12	2.72e+08	**1.40e-02**	2.28e+07
	Mean	8.81e-12	4.42e+02	3.30e-08	2.29e+12	2.45e+08	8.77e-03	1.10e+07
	Std	1.49e-12	9.94e+00	5.20e-09	9.97e+11	1.64e+07	**2.46e-03**	5.44e+06
MLCC	Best	**0.00e+00**	**1.73e-11**	1.28e-13	4.27e+12	2.15e+08	5.85e+06	4.16e+04
	Median	**0.00e+00**	**6.43e-11**	1.46e-13	1.03e+13	3.92e+08	1.95e+07	5.15e+05
	Worst	**3.83e-26**	**1.09e+01**	1.86e-11	1.62e+13	4.87e+08	1.98e+07	**2.78e+06**
	Mean	**1.53e-27**	**5.57e-01**	9.88e-13	9.61e+12	3.84e+08	1.62e+07	6.89e+05
	Std	**7.66e-27**	**2.21e+00**	3.70e-12	3.43e+12	6.93e+07	4.97e+06	**7.37e+05**
jDEsps	Best	2.63e-26	2.01e+01	**5.73e-14**	**2.62e+11**	**5.01e+07**	**4.00e-09**	**3.00e+02**
	Median	2.00e-23	8.20e+01	**1.03e-13**	**4.78e+11**	**7.99e+07**	**2.18e-08**	**7.94e+03**
	Worst	1.82e-22	3.29e+02	**4.48e-13**	**4.71e+12**	**9.27e+07**	1.34e-01	1.36e+07
	Mean	4.10e-23	1.10e+02	**1.30e-13**	**8.15e+11**	7.71e+07	**5.58e-03**	**5.77e+05**
	Std	4.77e-23	8.29e+01	**9.18e-14**	**9.46e+11**	1.09e+07	2.67e-02	2.72e+06
FEs		F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}
DECC-G	Best	6.37e+06	2.66e+08	1.03e+04	2.06e+01	7.78e+04	1.78e+03	6.96e+08
	Median	6.70e+07	3.18e+08	1.07e+04	2.33e+01	8.87e+04	3.00e+03	8.07e+08
	Worst	9.22e+07	3.87e+08	1.17e+04	2.79e+01	1.07e+05	1.66e+04	9.06e+08
	Mean	6.44e+07	3.21e+08	1.06e+04	2.34e+01	8.93e+04	5.12e+03	8.08e+08
	Std	2.89e+07	3.38e+07	2.95e+02	1.78e+00	6.87e+03	3.95e+03	6.07e+07
DECC-G*	Best	2.79e+07	1.18e+07	2.33e+03	5.82e-08	**6.16e+01**	3.78e+02	2.46e+07
	Median	4.07e+07	1.41e+07	2.49e+03	7.52e-08	**7.72e+01**	5.40e+02	2.90e+07
	Worst	1.50e+08	1.77e+07	2.64e+03	8.79e-01	**1.19e+02**	7.55e+02	3.56e+07
	Mean	6.14e+07	1.41e+07	2.48e+03	3.52e-02	**7.87e+01**	5.50e+02	2.91e+07
	Std	3.24e+07	1.39e+06	7.63e+01	1.76e-01	**1.41e+01**	**9.78e+01**	2.91e+06
MLCC	Best	4.51e+04	8.96e+07	2.52e+03	1.96e+02	2.42e+04	1.01e+03	2.62e+08
	Median	4.67e+07	1.24e+08	3.16e+03	1.98e+02	3.47e+04	1.91e+03	3.16e+08
	Worst	9.06e+07	1.46e+08	5.90e+03	1.98e+02	4.25e+04	3.47e+03	3.77e+08
	Mean	4.38e+07	1.23e+08	3.43e+03	1.98e+02	3.49e+04	2.08e+03	3.16e+08
	Std	3.45e+07	1.33e+07	8.72e+02	6.98e-01	4.92e+03	7.27e+02	2.77e+07
jDEsps	Best	**2.34e-19**	**9.11e+03**	**1.81e+03**	**1.58e-13**	3.01e+02	**2.27e-02**	**2.55e+04**
	Median	**2.01e-17**	**1.63e+04**	**1.83e+03**	**2.36e-13**	2.02e+03	**6.18e+01**	**3.15e+05**
	Worst	**1.13e+07**	**1.36e+05**	**1.97e+03**	**4.85e-04**	1.95e+05	**7.32e+02**	**1.57e+06**
	Mean	**1.52e+06**	**2.31e+04**	**1.85e+03**	**1.94e-05**	1.57e+04	**1.86e+02**	**3.85e+05**
	Std	**3.63e+06**	**2.50e+04**	**4.66e+01**	**9.70e-05**	4.05e+04	2.32e+02	**3.54e+05**
FEs		F_{15}	F_{16}	F_{17}	F_{18}	F_{19}	F_{20}	
DECC-G	Best	1.09e+04	5.97e+01	2.50e+05	5.61e+03	1.02e+06	3.59e+03	
	Median	1.18e+04	7.51e+01	2.89e+05	2.30e+04	1.11e+06	3.98e+03	
	Worst	1.39e+04	9.24e+01	3.26e+05	4.71e+04	**1.20e+06**	5.32e+03	
	Mean	1.22e+04	7.66e+01	2.87e+05	2.46e+04	1.11e+06	4.06e+03	
	Std	8.97e+02	8.14e+00	1.98e+04	1.05e+04	**5.15e+04**	3.66e+02	
DECC-G*	Best	**3.62e+03**	**7.04e-08**	**8.09e+01**	8.37e+02	9.90e+05	2.83e+03	
	Median	**3.88e+03**	**1.04e-07**	**1.03e+02**	1.08e+03	1.15e+06	3.21e+03	
	Worst	**4.25e+03**	**2.18e+00**	**1.33e+02**	**1.53e+03**	1.23e+06	6.23e+03	
	Mean	**3.88e+03**	**4.01e-01**	**1.03e+02**	1.08e+03	1.14e+06	3.33e+03	
	Std	**1.76e+02**	**6.59e-01**	**1.38e+01**	**1.61e+02**	5.85e+04	6.63e+02	
MLCC	Best	5.30e+03	2.08e+02	1.38e+05	2.51e+03	1.21e+06	1.70e+03	
	Median	6.89e+03	3.95e+02	1.59e+05	4.17e+03	1.36e+06	2.04e+03	
	Worst	1.04e+04	3.97e+02	1.86e+05	1.62e+04	1.54e+06	2.34e+03	
	Mean	7.11e+03	3.76e+02	1.59e+05	7.09e+03	1.36e+06	2.05e+03	
	Std	1.34e+03	4.71e+01	1.43e+04	4.77e+03	7.35e+04	**1.80e+02**	
jDEsps	Best	4.82e+03	3.57e+00	2.90e+03	**5.67e+02**	**5.19e+04**	**1.19e+02**	
	Median	5.48e+03	3.57e+00	9.95e+03	**9.20e+02**	**8.14e+05**	**9.70e+02**	
	Worst	6.31e+03	3.84e+01	4.52e+05	1.76e+03	1.37e+06	**1.61e+03**	
	Mean	5.50e+03	4.97e+00	5.52e+04	**9.73e+02**	**8.00e+05**	**8.79e+02**	
	Std	3.60e+02	6.97e+00	1.07e+05	2.86e+02	3.34e+05	4.87e+02	

B. Experimental Results

In the experiments, algorithm's parameters were set as follows:

- F was self-adaptive, $F_i^{(0)} = 0.5$,
- CR was self-adaptive, $CR_i^{(0)} = 0.9$,
- NP was changing, initial value $NP_{init} = 100$,
- $pmax = 3$ was fixed during the optimization.

$pmax$ value implies that population size is 20 after $pmax - 1$ population size reductions.

The obtained results (error values $f(\vec{x}) - f(\vec{x}^*)$) are presented in Table II. Table III shows the results of cooperative co-evolution algorithms from literature along with results of jDEsps algorithm when $FEs = 3.0e + 6$.

Convergence graphs of the jDEsps algorithm for eight selected benchmark functions are presented in Fig. 1.

PC Configure:
System: GNU/Linux, CPU: 2.5 GHz, RAM: 4 GB, Language: C/C++, Algorithm: jDEsps, Compiler: GNU Compiler (g++).

If we compare the proposed jDEsps with the jDElsgo [4] using overall scores as prescribed by organizing committee of CEC 2012 [15], jDEsps has 7171 points and outperformed jDElsgo with 5736 points (on $3e6$, 2166 and 1704, respectively).

VI. Conclusions

In this paper we presented the jDEsps algorithm. The performance of the algorithm was evaluated on the set of benchmark functions provided for CEC 2012 special session on large-scale global optimization.

The main characteristics of our algorithm are as follow: a self-adaptive control mechanism to change two DE's control parameters (F and CR) during the optimization process, small population size at early generations, bigger population size in later generation, and gradually reducing population size, more strategies, one strategy is responsible for preforming a local search, and, finally, a mechanism for changing the sign of F control parameter.

The experimental results give evidence that the jDEsps algorithm obtains the results that are competitive when comparing to other algorithms for high-dimensional continuous optimization problems.

Acknowledgment

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

References

[1] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[2] J. Brest and M. Sepesy Maučec. Population Size Reduction for the Differential Evolution Algorithm. *Applied Intelligence*, 29(3):228–247, 2008.

[3] J. Brest and M. S. Maučec. Self-adaptive differential evolution algorithm using population size reduction and three strategies. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2157–2174, 2011.

[4] J. Brest, A. Zamuda, B. Bošković, Iztok Fister, and M. S. Maučec. Large Scale Global Optimization using Self-adaptive Differential Evolution Algorithm. In *IEEE World Congress on Computational Intelligence*, pages 3097–3104, 2010.

[5] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer. High-dimensional Real-parameter Optimization Using Self-adaptive Differential Evolution Algorithm with Population Size Reduction. In *2008 IEEE World Congress on Computational Intelligence*, pages 2032–2039. IEEE Press, 2008.

[6] S. Das, A. Abraham, U.K. Chakraborty, and A. Konar. Differential Evolution Using a Neighborhood-based Mutation Operator. *IEEE Transactions on Evolutionary Computation*, 13(3):526–553, 2009.

[7] S. Das and P.N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):27–54, 2011.

[8] C. Garca-Martnez, F. Rodrguez, and M. Lozano. Role differentiation and malleable mating for differential evolution: an analysis on large-scale optimisation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2109–2126, 2011.

[9] A. LaTorre, S. Muelas, and J.-M. Peña. A mos-based dynamic memetic differential evolution algorithm for continuous optimization: a scalability test. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2187–2199, 2011.

[10] M. Lozano, D. Molina, and F. Herrera. Editorial scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2085–2087, 2011.

[11] R. Mallipeddi and P.N. Suganthan. Ensemble of Constraint Handling Techniques. *IEEE Transactions on Evolutionary Computation*, 14(4):561–579, 2010.

[12] F. Neri and V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, 33(1–2):61–106, 2010.

[13] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2):398–417, 2009.

[14] R. Storn and K. Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[15] K. Tang, Xiaodong Li, P. N. Suganthan, Z. Yang, and T. Weise. 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, 2009.

[16] N. S. Teng, J. Teo, and M. H. A. Hijazi. Self-adaptive population sizing for a tune-free differential evolution. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 13(7):709–724, 2009.

[17] H. Wang, Z. Wu, and S. Rahnamayan. Enhanced opposition-based differential evolution for solving high-dimensional continuous optimization problems. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2127–2140, 2011.

[18] M. Weber, F. Neri, and V. Tirronen. Shuffle or update parallel differential evolution for large-scale optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2089–2107, 2011.

[19] Z. Yang, K. Tang, and X. Yao. Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2141–2155, 2011.

[20] J. Zhang and A.C. Sanderson. JADE: Adaptive Differential Evolution with Optional External Archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.

[21] S.-Z. Zhao, N. Suganthan, P. and S. Das. Self-adaptive Differential Evolution with Modified Multi-Trajectory Search for CEC2010 Large Scale Optimization. In Bijaya Panigrahi, Swagatam Das, Ponnuthurai Suganthan, and Subhransu Dash, editors, *Swarm, Evolutionary, and Memetic Computing*, volume 6466 of *Lecture Notes in Computer Science*, pages 1–10. Springer Berlin / Heidelberg, 2010.

[22] S.-Z. Zhao, P. N. Suganthan, and S. Das. Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15(11):2175–2185, 2011.