# Real Parameter Single Objective Optimization using Self-Adaptive Differential Evolution Algorithm with more Strategies

Janez Brest, Borko Bošković, Aleš Zamuda, Iztok Fister
Institute of Computer Science,
Faculty of Electrical Engineering and Computer Science,
University of Maribor,
Smetanova ul. 17, 2000 Maribor, Slovenia
Email: janez.brest@uni-mb.si

Efrén Mezura-Montes
Departamento de Inteligencia Artificial
Universidad Veracruzana
Sebastián Camacho 5, Centro,
Xalapa, Veracruz, 91000, MEXICO
Email: emezura@uv.mx

***Abstract*—A new differential evolution algorithm for single objective optimization is presented in this paper. The proposed algorithm uses a self-adaptation mechanism for parameter control, divides its population into more subpopulations, applies more DE strategies, promotes population diversity, and eliminates the individuals that are not changed during some generations. The experimental results obtained by our algorithm on the benchmark consisting of 25 test functions with dimensions $D = 10$, $D = 30$, and $D = 50$ as provided for the CEC 2013 competition and special session on Real Parameter Single Objective Optimization are presented.**

Key words: single objective optimization, differential evolution, self-adaptation.

## I. Introduction

This paper presents an algorithm for Real Parameter Single Objective Optimization (RPSOO) [1]. Single objective optimization algorithms serve as a basis of the more complex optimization algorithms such as multi-objective optimizations algorithms, niching algorithms, constrained optimization algorithms, and so on [1].

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

Differential Evolution (DE) [2], [3], [4] is a well-known evolutionary algorithm for optimization in continuous and discrete domains [5], [6]. Recently, in the literature, it can be found that this algorithm is highly competitive, especially in areas, where real-parameters are needed [7], [8], [9], [10].

DE is a population based evolutionary algorithms. Its initial population is generated at random with a Uniform distribution over the entire search space. Then DE repeatedly applies mutation, crossover, and selection operators over the population to generate a next population. The evolutionary process stops when stopping criteria is met and then the DE algorithm reports the best solution found.

Three control parameters are used in the original DE, proposed by Storn and Price in 1995: the amplification factor of the difference vector $F$, the crossover control parameter $CR$, and population size $NP$. All three control parameters are fixed during the optimization process in the original DE algorithm. A tuning of the control parameters can be used to find out good values for them before the actual optimization process starts. A tuning process might be time consuming, especially, when it is performed as hand-tuning. In order to overcome the tuning problems and to improve algorithm's performance, adaptive and self-adaptive mechanisms were applied on the DE control parameters [4], [3].

In the specialized 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 in [11], [12]. Both self-adaptive techniques, known also as algorithms *jDE* and $SaDE$, have a great research influence onto DE-based algorithms. Moreover, many other mechanisms have been proposed [13]. Some years ago only a few researches were related with the third DE control parameter, i.e., population size. But more recently, works dealing with population size i.e. changing population size during the evolutionary process are increasing [14]. They outlined that $NP$ plays also an important role among control parameters in the DE.

A self-adaptive *jDE* algorithm was introduced in 2006 [11]. jDE-based algorithms were applied to solve large-scale single objective optimization problems: CEC 2008 [15], CEC 2010 [16], CEC 2012 [17], large-scale continuous optimization problems [18]. However, various mechanisms are proposed in order to changing the DE control parameters adaptively or self-adaptively [19], [20], [17].

Qin and Suganthan in [12] proposed Self-adaptive Differential Evolution algorithm (SaDE), where the choice of a learning strategy and the two control parameters $F$ and $CR$ are gradually self-adapted according to the learning experience. In the specialized literature, usually, control parameters $F$ and $CR$ are changing in an adaptive or self-adaptive manner [3], [21], [4].

In this paper, an evaluation of our self-adaptive $jDEsoo$ algorithm is performed on the benchmark functions provided for the CEC 2013 special session on RPSOO [1], i.e., a successor of special Session on Real-Parameter Optimization that has been occurred on CEC 2005 [22]. The DE-based algorithms presented at this competition, like "DE" [23] and "L-SADE" [24], were one of the more competitive among other algorithms in such previous competition [25].

The structure of the paper is as follows. Section II gives background for this work, where an overview of DE, a description of the self-adaptive control parameters, and outlines of our previous algorithm are given. Section III presents a new variant of the algorithm, called $jDEsoo$. In Section IV experimental results of the $jDEsoo$ algorithm on single objective benchmark functions are presented. Section V concludes the paper with some final remarks.

## II. Background

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

### A. The Differential Evolution

DE is a population-based algorithm. It uses mutation, crossover and selection operators to generate a next population from the current population.

Let us briefly explain the original DE algorithm [2]. The population in generation $G$ consists of $NP$ vectors:

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

*1) Mutation:* A mutant vector $\vec{v}_i^{(G)}$ is created by using one of the DE mutation strategies. The most popular 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 the set $\{1, NP\}$ and $r_1 \neq r_2 \neq r_3 \neq i$. $F$ is a mutation scale factor within the range $[0, 2]$, usually less than 1. $\vec{x}_{best,G}$ denotes the best vector in generation $G$.

The other useful DE strategies [3], [4] are:

- "rand/1": $\vec{v}_{i,G} = \vec{x}_{r_1,G} + F(\vec{x}_{r_2,G} - \vec{x}_{r_3,G})$,
- "best/1": $\vec{v}_{i,G} = \vec{x}_{best,G} + F(\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$,
- "current to best/1": $\vec{v}_{i,G} = \vec{x}_{i,G} + F(\vec{x}_{best,G} - \vec{x}_{i,G}) + F(\vec{x}_{r_1,G} - \vec{x}_{r_2,G})$,
- "best/2": $\vec{v}_{i,G} = \vec{x}_{best,G} + F(\vec{x}_{r_1,G} - \vec{x}_{r_2,G}) + F(\vec{x}_{r_3,G} - \vec{x}_{r_4,G})$,
- "rand/2": $\vec{v}_{i,G} = \vec{x}_{r_1,G} + F(\vec{x}_{r_2,G} - \vec{x}_{r_3,G}) + F(\vec{x}_{r_4,G} - \vec{x}_{r_5,G})$,

where the indexes $r_1$–$r_5$ represent the random and mutually different integers generated within the range $\{1, NP\}$ and also different from index $i$.

*2) Crossover:* A crossover operator forms a trial vector $\vec{u}_i^{(G)}$ as follows:

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

for $i = 1, 2, ..., NP$ and $j = 1, 2, ..., D$. The 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.

If the control parameters from the trial vector are out of bounds, the proposed solutions found in the literature [2], [26] are: they are reflected into bounds, set on bounds or used as they are (out of bounds).

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

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

The DE has a greedy selection, while other evolutionary algorithms have a more sophisticate selection operation. Fitness values between the population vector and its corresponding trial vector are compared in DE. This better vector will survive and become a member of the population for the next generation.

### B. Self-Adaptation of F and CR

The jDE algorithm [11] uses a 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 operator as follows [11]:

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

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

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

In [11] parameters $\tau_1, \tau_2, F_l, F_u$ are fixed to values $0.1, 0.1, 0.1, 0.9$, respectively.

## III. A new Variant of the Algorithm

In this section our new algorithm $jDEsoo$ for solving real-parameter single objective optimization is presented. The $jDEsoo$ algorithm includes some good characteristics and mechanisms that are similar to our previous algorithms jDElsgo [17] and jDElscop [18] that were used for solving large-scale global optimization ($D$ was upto 1000).

The pseudo-code of new $jDEsoo$ algorithm is presented in Algorithm 1. The idea of how to divide a population into sub-population is depicted in Figure 1.

The new proposed algorithm uses *jDE* as a basis and updated it with the following features:

- Dividing the whole population into subpopulations,
- Applying a different DE strategy in each subpopulation,

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: $NP = 30$
9: $itAge = 1000$
10:
11: {** Generate uniformly distributed random population within search space **}
12: **for** $(it = 0;\ it < MaxFEs;\ it = it + 1)$ **do**
13: $i = it$ **mod** $NP$ {mod ... modulo operation}
14:
15: {** Perform one iteration of the jDE using one of three strategies. BIN crossover is used. **}
16:
17: **if** $(rand(0,1) < 0.5$ **and** $i < b_1)$ **then**
18: {** jDEbest ** }
19: $F_l = 0.1 + \sqrt{\frac{1}{NP}};\ F_u = 1.0;\ \ CR_l = 0.7;\ CR_u = 0.9;$
20: $\vec{u}$ =jDEbest($imin$, $pop$)
21: **else**
22: **if** $(i < b_2)$ **then**
23: {** jDEbin **}
24: $F_l = 0.1;\ F_u = 1.0;\ \ CR_l = 0.0;\ CR_u = 1.0 - \frac{1.0}{NP};$
25: $\vec{u}$ =jDEbin($i$, $pop$)
26: **else**
27: {** jDEbinSec **}
28: $F_l = 0.8;\ F_u = 1.0;\ \ CR_l = 0.8;\ CR_u = 1.0 - \frac{1.0}{NP};$
29: $r_1 = 0.7 * NP + 0.3 * NP * rand(0,1);$
30: $\vec{u}$ =jDEbinSec($i$, $pop$)
31: **end if**
32: **end if**
33: {Bound check; if volatile then set on bound}
34: $f(\vec{u})$ {Function evaluation}
35: {Selection}
36:
37: {** apply aging at every $itAge$ iteration **}
38: **if** ($it$ **mod** $itAge == (itAge - 1)$) **then**
39: {for every individual test if its age is greater tha 50 then reinitialize is with probability $0.1$}
40: **end if**
41: **end for**

**Algorithm 1:** *jDEsoo* algorithm

- Ensuring the population diversity during generations,
- Aging mechanism.

The initial population is generated at random with a Uniform distribution between the lower $x_{j,low}$ and upper $x_{j,upp}$ bounds defined for each variable $x_j$.

If an individual was not improved for some generations (we set the value for checking this every 1000 iterations – it is approx. 33 generations) then it is reinitialized with probability of $0.1$. We set the value for aging to 50. Please note that every 1000 iterations the whole population is tested if aging criteria is fulfill.

In this paper we used three simple DE strategies "DE/rand/1/best", "DE/rand/1/bin", and, once again "DE/rand/1/bin", and we set $b_1 = 0.1 * NP$ and $b_2 = 0.9 * NP$. The strategies used their own self-adaptive control parameters $F$ and $CR$. They have different bound values of $F_l$, $F_u$, $CR_l$, and $CR_u$ (see Algorithm 1), which can be calculated outside the iteration loop since they are constant in the whole search process.

The reasons of employing "DE/rand/1/best", "DE/rand/1/bin" strategies were made based on our previous experiences and the suggestion from literature. The "best" strategy has fast convergence ability (exploitation), while "DE/rand/1/bin" has good exploration ability.

We used "DE/rand/1/bin" strategy two-times, one with wide range of $CR$ parameter, while the second with higher value for $CR$ and also $F$ ($CR_l = 0.8$, $F_l = 0.8$).

TABLE I. PROPERTIES OF THE CEC 2013 BENCHMARK FUNCTIONS [1]

| | No. | Functions | $f_i^* = f_i(x^*)$ |
|---|---|---|---|
| Unimodal Functions | 1 | Sphere Function | -1400 |
| | 2 | Rotated High Conditioned Elliptic Function | -1300 |
| | 3 | Rotated Bent Cigar Function | -1200 |
| | 4 | Rotated Discus Function | -1100 |
| | 5 | Different Powers Function | -1000 |
| Basic Multimodal Functions | 6 | Rotated Rosenbrock's Function | -900 |
| | 7 | Rotated Schaffers F7 Function | -800 |
| | 8 | Rotated Ackley's Function | -700 |
| | 9 | Rotated Weierstrass Function | -600 |
| | 10 | Rotated Griewank's Function | -500 |
| | 11 | Rastrigin's Function | -400 |
| | 12 | Rotated Rastrigin's Function | -300 |
| | 13 | Non-Continuous Rotated Rastrigin's Function | -200 |
| | 14 | Schwefel's Function | -100 |
| | 15 | Rotated Schwefel's Function | 100 |
| | 16 | Rotated Katsuura Function | 200 |
| | 17 | Lunacek Bi_Rastrigin Function | 300 |
| | 18 | Rotated Lunacek Bi_Rastrigin Function | 400 |
| | 19 | Expanded Griewank's plus Rosenbrock's Function | 500 |
| | 20 | Expanded Scaffer's F6 Function | 600 |
| Composition Functions | 21 | Composition Function 1 (n=5, Rotated) | 700 |
| | 22 | Composition Function 2 (n=3, Unrotated) | 800 |
| | 23 | Composition Function 3 (n=3, Rotated) | 900 |
| | 24 | Composition Function 4 (n=3, Rotated) | 1000 |
| | 25 | Composition Function 5 (n=3, Rotated) | 1100 |
| | 26 | Composition Function 6 (n=5, Rotated) | 1200 |
| | 27 | Composition Function 7 (n=5, Rotated) | 1300 |
| | 28 | Composition Function 8 (n=5, Rotated) | 1400 |
| Search Range: $[-100,100]^D$ | | | |

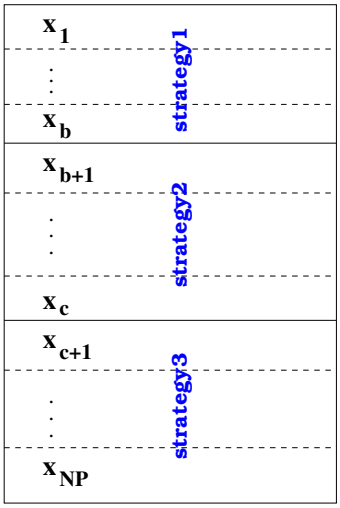


Fig. 1. Population is divided into more sub-population. Each sub-population uses its DE strategy.

## IV. RESULTS

### A. Benchmark Functions

The *jDEsoo* algorithm was tested on a set of 28 benchmark functions [1]. The benchmark functions are scalable. The dimensions of benchmark functions were $D = 10$, $D = 30$, and $D = 50$, respectively, and 51 runs of algorithm were needed for each function. The optimal values are known for all benchmark functions. We used our algorithm as black-box optimizer as required for this special session.

The general features of these functions are presented in Table I.

### B. Experimental Results

Notice, that in this competition, error values smaller than $10^{-8}$ are taken as zero.

In the experiments, parameters of the *jDEsoo* algorithm were set as follows:

- $F$ was self-adaptive,

TABLE II. EXPERIMENTAL RESULTS WITH DIMENSION $D = 10$.

| Func. | Best | Worst | Median | Mean | Std |
|---|---|---|---|---|---|
| 1 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 |
| 2 | 1.0250e+01 | 6.1157e+03 | 1.2819e+03 | 1.7180e+03 | 1.7072e+03 |
| 3 | 0.0000e+00 | 1.0375e+01 | 2.9987e-03 | 1.6071e+00 | 2.9930e+00 |
| 4 | 2.3937e-06 | 2.3242e+00 | 9.5699e-03 | 1.2429e-01 | 3.8021e-01 |
| 5 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 |
| 6 | 0.0000e+00 | 9.8124e+00 | 9.8124e+00 | 8.4982e+00 | 3.3348e+00 |
| 7 | 7.4725e-05 | 2.1685e+01 | 6.9981e-02 | 9.4791e-01 | 3.2840e+00 |
| 8 | 2.0174e+01 | 2.0498e+01 | 2.0355e+01 | 2.0348e+01 | 7.6220e-02 |
| 9 | 2.8689e-01 | 6.0739e+00 | 2.4698e+00 | 2.7464e+00 | 1.3910e+00 |
| 10 | 7.3960e-03 | 1.5501e-01 | 6.6444e-02 | 7.0960e-02 | 3.5194e-02 |
| 11 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 |
| 12 | 1.9899e+00 | 1.7909e+01 | 4.9748e+00 | 6.1144e+00 | 3.2590e+00 |
| 13 | 9.9496e-01 | 2.0170e+01 | 6.6802e+00 | 7.8102e+00 | 4.8278e+00 |
| 14 | 0.0000e+00 | 1.8736e-01 | 6.2454e-02 | 5.0208e-02 | 5.7267e-02 |
| 15 | 2.1581e+02 | 1.2673e+03 | 8.9024e+02 | 8.4017e+02 | 2.3364e+02 |
| 16 | 5.9909e-01 | 1.5403e+00 | 1.1141e+00 | 1.0991e+00 | 2.3476e-01 |
| 17 | 0.0000e+00 | 1.0125e+01 | 1.0122e+01 | 9.9240e+00 | 1.4174e+00 |
| 18 | 1.8121e+01 | 3.8264e+01 | 2.7396e+01 | 2.7716e+01 | 5.4304e+00 |
| 19 | 0.0000e+00 | 5.5773e-01 | 3.3110e-01 | 3.1993e-01 | 1.0542e-01 |
| 20 | 1.8668e+00 | 3.5941e+00 | 2.7428e+00 | 2.7178e+00 | 4.8839e-01 |
| 21 | 1.0000e+02 | 4.0019e+02 | 4.0019e+02 | 3.5113e+02 | 9.0354e+01 |
| 22 | 5.0955e+00 | 1.1563e+02 | 1.0180e+02 | 9.1879e+01 | 2.9565e+01 |
| 23 | 4.2360e+02 | 1.2791e+03 | 8.1673e+02 | 8.1116e+02 | 2.2219e+02 |
| 24 | 1.1476e+02 | 2.1812e+02 | 2.1084e+02 | 2.0851e+02 | 1.3831e+01 |
| 25 | 2.0000e+02 | 2.1873e+02 | 2.0968e+02 | 2.0955e+02 | 4.3260e+00 |
| 26 | 1.0398e+02 | 3.1479e+02 | 2.0002e+02 | 1.9301e+02 | 4.3758e+01 |
| 27 | 3.0000e+02 | 5.5239e+02 | 5.0799e+02 | 4.9412e+02 | 5.2492e+01 |
| 28 | 1.0000e+02 | 3.0000e+02 | 3.0000e+02 | 2.8824e+02 | 4.7527e+01 |

- $CR$ was self-adaptive,
- $NP$ was fixed during the optimization process, $NP_{init} = 30$,
- aging was set to 50.

The obtained results (error values $f(\vec{x}) - f(\vec{x}^*)$) are presented in Tables II, III, IV for dimensions 10, 30, and 50, respectively.

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

TABLE V. COMPUTATIONAL COMPLEXITY

| $D$ | $T0$ | $T1$ | $T2$ | $(T2 - T1)/T0$ |
|---|---|---|---|---|
| 10 | 0.07966 $s$ | 0.6259 $s$ | 0.6894 $s$ | 0.7967 |
| 30 | | 1.8745 $s$ | 1.9893 $s$ | 1.4404 |
| 50 | | 3.1391 $s$ | 3.3101 $s$ | 2.1456 |

The algorithm complexity is presented in Table V as required in [1].

## V. CONCLUSIONS

The $jDEsoo$ algorithm was presented in this paper. The performance of the algorithm was evaluated on the set of benchmark functions provided for CEC 2013 special session on real-parameter single objective optimization.

The algorithm uses a relatively small population size that remained fixed for all 28 benchmark functions and dimensions ($D = 10$, $D = 30$, $D = 50$).

The performance of this algorithm against other algorithm will be performed by organisers of this special session.

## ACKNOWLEDGMENT

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

## REFERENCES

[1] J. J. Liang, B.-Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, Tech. Rep. 201212, 2013. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/

TABLE III. EXPERIMENTAL RESULTS WITH DIMENSION $D = 30$.

| Func. | Best | Worst | Median | Mean | Std |
|---|---|---|---|---|---|
| 1 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 |
| 2 | 2.1829e+04 | 4.7209e+05 | 9.7478e+04 | 1.2914e+05 | 9.6860e+04 |
| 3 | 4.9775e+01 | 6.6837e+07 | 6.7877e+05 | 9.8414e+06 | 1.8562e+07 |
| 4 | 8.2829e+00 | 4.2142e+04 | 2.1112e+04 | 1.9720e+04 | 1.2605e+04 |
| 5 | 0.0000e+00 | 6.6224e-08 | 1.4148e-08 | 1.2606e-08 | 1.3669e-08 |
| 6 | 1.5361e+00 | 2.6407e+01 | 5.5009e+00 | 7.9292e+00 | 7.5864e+00 |
| 7 | 4.4141e-01 | 2.5565e+01 | 8.6761e+00 | 9.8167e+00 | 6.4998e+00 |
| 8 | 2.0842e+01 | 2.1017e+01 | 2.0954e+01 | 2.0946e+01 | 4.5129e-02 |
| 9 | 1.1449e+01 | 3.3424e+01 | 1.8048e+01 | 2.0971e+01 | 7.1839e+00 |
| 10 | 1.2321e-02 | 2.1443e-01 | 6.6478e-02 | 7.9055e-02 | 4.3518e-02 |
| 11 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 | 0.0000e+00 |
| 12 | 1.4924e+01 | 8.0458e+01 | 4.1240e+01 | 4.2835e+01 | 1.5735e+01 |
| 13 | 2.5607e+01 | 1.3102e+02 | 7.1528e+01 | 7.0750e+01 | 2.3853e+01 |
| 14 | 4.1713e-02 | 5.1442e+00 | 1.2008e+00 | 1.3327e+00 | 1.3500e+00 |
| 15 | 2.8673e+03 | 6.2279e+03 | 4.8656e+03 | 4.8340e+03 | 5.9818e+02 |
| 16 | 1.4547e+00 | 2.8891e+00 | 2.3067e+00 | 2.2791e+00 | 3.6845e-01 |
| 17 | 3.0434e+01 | 3.0436e+01 | 3.0434e+01 | 3.0434e+01 | 4.4649e-04 |
| 18 | 8.9001e+01 | 1.8493e+02 | 1.2096e+02 | 1.2341e+02 | 1.8512e+01 |
| 19 | 4.1551e-01 | 1.5554e+00 | 1.1236e+00 | 1.0956e+00 | 2.8552e-01 |
| 20 | 1.0510e+01 | 1.2513e+01 | 1.1664e+01 | 1.1639e+01 | 4.4082e-01 |
| 21 | 2.0000e+02 | 4.4354e+02 | 3.0000e+02 | 2.9396e+02 | 8.2895e+01 |
| 22 | 6.7721e+00 | 2.2164e+02 | 1.3832e+01 | 5.1621e+01 | 5.7787e+01 |
| 23 | 3.6845e+03 | 6.2774e+03 | 4.5862e+03 | 4.6061e+03 | 5.4328e+02 |
| 24 | 2.3298e+02 | 2.6760e+02 | 2.4777e+02 | 2.4818e+02 | 7.4756e+00 |
| 25 | 2.4698e+02 | 2.7437e+02 | 2.5977e+02 | 2.6037e+02 | 6.8567e+00 |
| 26 | 2.0000e+02 | 3.5318e+02 | 2.0002e+02 | 2.5758e+02 | 6.9612e+01 |
| 27 | 5.0522e+02 | 1.0349e+03 | 7.1076e+02 | 7.2161e+02 | 8.6905e+01 |
| 28 | 3.0000e+02 | 3.0000e+02 | 3.0000e+02 | 3.0000e+02 | 3.8343e-04 |


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

[3] S. Das and P. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 27–54, 2011.

[4] F. Neri and V. Tirronen, "Recent advances in differential evolution: a survey and experimental analysis," *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 61–106, 2010.

[5] I. Fister and J. Brest, "Using differential evolution for the graph coloring," in *IEEE SSCI2011 symposium series on computational intelligence. Piscataway: IEEE*, 2011, pp. 150–156.

[6] I. Fister, M. Mernik, and B. Filipič, "Graph 3-coloring with a hybrid self-adaptive evolutionary algorithm," *Computational Optimization and Applications*, vol. 54, pp. 741–770, 2013.

[7] S. Amirabadi, S. Kabiri, R. Vakili, D. Iranshahi, and M. R. Rahimpour, "Differential Evolution Strategy for Optimization of Hydrogen Production via Coupling of Methylcyclohexane Dehydrogenation Reaction and Methanol Synthesis Process in a Thermally Coupled Double Membrane Reactor," *Industrial & Engineering Chemistry Research*, vol. 52, no. 4, pp. 1508–1522, JAN 30 2013.

[8] K. Opara and J. Arabas, "Decomposition and Metaoptimization of Mutation Operator in Differential Evolution," in *Swarm and Evolutionary Computation*, ser. Lecture Notes in Computer Science, Rutkowski, L and Korytkowski, M and Scherer, R and Tadeusiewicz, R and Zadeh, LA and Zurada, JM, Ed., vol. 7269, 2012, Proceedings Paper, pp. 110–118.

[9] R. Polakova and J. Tvrdik, "A Comparison of Two Adaptation Approaches in Differential Evolution," in *Swarm and Evolutionary Computation*, ser. Lecture Notes in Computer Science, Rutkowski, L and Korytkowski, M and Scherer, R and Tadeusiewicz, R and Zadeh, LA and Zurada, JM, Ed., vol. 7269, 2012, Proceedings Paper, pp. 317–324.

[10] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "An Improved Self-Adaptive Differential Evolution Algorithm for Optimization Problems," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 89–99, FEB 2013.

[11] 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*, vol. 10, no. 6, pp. 646–657, 2006.

[12] 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*, vol. 13, no. 2, pp. 398–417, 2009.

[13] M. Weber, F. Neri, and V. Tirronen, "Shuffle or update parallel differential evolution for large-scale optimization," *Soft Computing*, vol. 15, pp. 2089–2107, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00500-010-0640-9

[14] I. Fajfar, T. Tuma, J. Puhan, J. Olensek, and A. Burmen, "Towards Smaller Populations in Differential Evolution," *INFORMACIJE MIDEM-JOURNAL OF MICROELECTRONICS ELECTRONIC COMPONENTS AND MATERIALS*, vol. 42, no. 3, pp. 152–163, 2012.

[15] 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*. IEEE Press, 2008, pp. 2032–2039.

[16] J. Brest, A. Zamuda, B. Bošković, I. Fister, and M. S. Maučec, "Large

TABLE IV. EXPERIMENTAL RESULTS WITH DIMENSION $D = 50$.

| Func. | Best | Worst | Median | Mean | Std |
|---|---|---|---|---|---|
| 1 | 0.0000e+00 | 1.4645e-07 | 1.8271e-08 | 2.7646e-08 | 3.2481e-08 |
| 2 | 2.3525e+05 | 1.2237e+06 | 5.6911e+05 | 6.0545e+05 | 2.4567e+05 |
| 3 | 8.6987e+04 | 3.5024e+08 | 1.7982e+07 | 4.7824e+07 | 6.8612e+07 |
| 4 | 5.0870e+04 | 1.3009e+05 | 8.1427e+04 | 8.3433e+04 | 1.5598e+04 |
| 5 | 5.9782e-07 | 7.0419e-06 | 2.3934e-06 | 2.4273e-06 | 1.1419e-06 |
| 6 | 2.5528e+01 | 4.9143e+01 | 4.3447e+01 | 4.2973e+01 | 3.7057e+00 |
| 7 | 1.3920e+01 | 6.5427e+01 | 2.4300e+01 | 2.9422e+01 | 1.2914e+01 |
| 8 | 2.1006e+01 | 2.1183e+01 | 2.1134e+01 | 2.1128e+01 | 3.8373e-02 |
| 9 | 2.3027e+01 | 6.3231e+01 | 5.6994e+01 | 5.3298e+01 | 9.7771e+00 |
| 10 | 1.2321e-02 | 3.2541e-01 | 1.4531e-01 | 1.4733e-01 | 7.6564e-02 |
| 11 | 0.0000e+00 | 9.9496e-01 | 4.6374e-08 | 1.9509e-02 | 1.3932e-01 |
| 12 | 4.8753e+01 | 1.6183e+02 | 9.6713e+01 | 9.7158e+01 | 2.5595e+01 |
| 13 | 1.1754e+02 | 2.2068e+02 | 1.7766e+02 | 1.7587e+02 | 2.3660e+01 |
| 14 | 1.1243e-01 | 3.5792e+01 | 6.8188e+00 | 8.0144e+00 | 6.7616e+00 |
| 15 | 7.5140e+03 | 1.1744e+04 | 9.4353e+03 | 9.4809e+03 | 1.0644e+03 |
| 16 | 2.1711e+00 | 3.8121e+00 | 3.2188e+00 | 3.1319e+00 | 3.9455e-01 |
| 17 | 5.0786e+01 | 5.3106e+01 | 5.0786e+01 | 5.0838e+01 | 3.2453e-01 |
| 18 | 1.5754e+02 | 2.7835e+02 | 2.1346e+02 | 2.1822e+02 | 3.1133e+01 |
| 19 | 1.1875e+00 | 3.6497e+00 | 2.2969e+00 | 2.2386e+00 | 5.4646e-01 |
| 20 | 2.0513e+01 | 2.2349e+01 | 2.1570e+01 | 2.1508e+01 | 4.3155e-01 |
| 21 | 2.0000e+02 | 1.1224e+03 | 1.1222e+03 | 8.2422e+02 | 4.0078e+02 |
| 22 | 1.3175e+01 | 2.2753e+02 | 1.9557e+01 | 3.0965e+01 | 3.8917e+01 |
| 23 | 6.8555e+03 | 1.1542e+04 | 9.5065e+03 | 9.4753e+03 | 1.0201e+03 |
| 24 | 2.6361e+02 | 3.1581e+02 | 2.8972e+02 | 2.8856e+02 | 1.2019e+01 |
| 25 | 2.9522e+02 | 3.3918e+02 | 3.1551e+02 | 3.1673e+02 | 1.0932e+01 |
| 26 | 3.7091e+02 | 4.5711e+02 | 3.8699e+02 | 3.9708e+02 | 2.3557e+01 |
| 27 | 9.4645e+02 | 1.4860e+03 | 1.1375e+03 | 1.1633e+03 | 1.2363e+02 |
| 28 | 4.0000e+02 | 3.5952e+03 | 4.0000e+02 | 9.4335e+02 | 1.1857e+03 |


Scale Global Optimization using Self-adaptive Differential Evolution Algorithm," in *IEEE World Congress on Computational Intelligence*, 2010, pp. 3097–3104.

[17] J. Brest, B. Bošković, A. Zamuda, I. Fister, and M. S. Maučec, "Self-Adaptive Differential Evolution Algorithm with a Small and Varying Population Size," in *2012 IEEE World Congress on Computational Intelligence (IEEE WCCI 2012), Brisbane, Australia*, 2012, pp. 2827–2834.

[18] J. Brest and M. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 15, no. 11, pp. 2157–2174, 2011.

[19] L.-A. Gordián-Rivera and E. Mezura-Montes, "A combination of specialized differential evolution variants for constrained optimization," in *Advances in Artificial Intelligence–IBERAMIA 2012*. Springer, 2012, pp. 261–270.

[20] A. Zamuda and J. Brest, "Population Reduction Differential Evolution with Multiple Mutation Strategies in Real World Industry Challenges," in *Swarm and Evolutionary Computation*, ser. Lecture Notes in Computer Science, Rutkowski, L and Korytkowski, M and Scherer, R and Tadeusiewicz, R and Zadeh, LA and Zurada, JM, Ed., vol. 7269, 2012, Proceedings Paper, pp. 154–161.

[21] A. W. Mohamed, H. Z. Sabry, and T. Abd-Elaziz, "Real parameter optimization by an effective differential evolution algorithm," *Egyptian Informatics Journal*, no. 0, pp. –, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1110866513000029

[22] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari, "Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization," Nanyang Technological University, Singapore and IIT Kanpur, India, Tech. Rep. #2005005, 2005. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan/

[23] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-Parameter Optimization with Differential Evolution," in *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol. 1. IEEE Press, Sept. 2005, pp. 506 – 513.

[24] A. K. Qin and P. N. Suganthan, "Self-adaptive Differential Evolution Algorithm for Numerical Optimization," in *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, vol. 2. IEEE Press, Sept. 2005, pp. 1785–1791.

[25] N. Hansen, "Compilation of Results on the CEC Benchmark Function Set," 2005. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/index_files/CEC\-05/compareresults.pdf

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