Self-adaptive Differential Evolution Algorithm with Population Size Reduction for Single Objective Bound-Constrained Optimization: Algorithm j21

Janez Brest
University of Maribor, FEECS
Maribor, Slovenia
Email: janez.brest@um.si

Mirjam Sepesy Maučec
University of Maribor, FEECS
Maribor, Slovenia
Email: mirjam.sepesy@um.si

Borko Bošković
University of Maribor, FEECS
Maribor, Slovenia
Email: borko.boskovic@um.si

***Abstract*—In this paper, we propose a new algorithm for solving real parameter single-objective optimization problems that were prepared for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Single-objective optimization problems are often very complex and computationally expensive. The presented algorithm, called j21, uses several mechanisms: two populations, vectors are chosen from both sub-populations in the mutation operation, crowding in the big population, population size reduction, etc. We show the experimental results for each benchmark function for two scenarios of different dimensions and eight configuration scenarios as required by the organizers of the CEC 2021 Special Session. We also compare the obtained results of j21 in a scenario with larger dimension and on one selected configuration with the original DE and j2020 algorithms.**

I. Introduction

Single-objective real parameter optimization can be found in many practical problems in different research domains. We are interested in optimizing one objective only, but this optimization can be a very challenging task since the search space is huge even for a small number of parameters, and it grows tremendously when the number of parameters is increasing [1].

Also, at CEC, the competition and/or special session on the Single-objective real parameter optimization has been organized in 2005 and in each year after 2013, and recently the competition has been extended also to the GECCO conference.

In this paper, we will tackle the optimization problems that are required to be minimized. A global optimization problem can be described as an objective function $f(\vec{x})$, where a vector $\vec{x} = \{x_1, x_2, ..., x_D\}$ consists of D variables or parameters. A goal in global optimization is to find a vector $\vec{x}$, called a solution, which minimizes the objective function $f(\vec{x})$. Each solution's variable $x_j, j = 1, 2, ..., D$ is defined by its lower $x_{j,low}$ and upper $x_{j,upp}$ bound. D denotes a dimension of the optimization problem.

An objective function might have many optima, and we are the most interested in cases where the algorithm finds the global one. If an algorithm traps itself into a local optimum, the obtained result of optimization may be poor [1].

Population-based algorithms, like evolutionary algorithms, swarm-based algorithms, etc., are suitable for solving real parameter optimization problems since they do not need any gradient calculation and have a good convergence speed toward a global optimum.

The Differential Evolution (DE) [2] algorithm was proposed by R. Storn and K. Price in 1995 as a technical report and in 1997 as a journal paper. We can put it in the group of evolutionary algorithms, and it has shown competitive performances when solving real-world optimization problems [3], [4], [5] in various domains. In recent years, a lot of improvements on DE were proposed, and they have been used in many practical works. We refer the interested readers to recently published reviews and surveys [6], [7], [8], [9], [10].

At the CEC 2019 competition on the 100-Digit Challenge [11], and the CEC 2020 competition on single-objective real parameter optimization some DE versions, more precisely DE version based on jDE [12], [13], were among the top algorithms [14], [15].

The presented algorithm, called j21, is based on the self-adaptive jDE100 [12] and j2020 [13] algorithms. It uses two populations, a restart mechanism that is applied in both populations to have a sufficient population diversity, a crowding mechanism, and a mechanism to choose vectors in the mutation operation from both sub-populations. These mechanisms are similar to mechanisms in the jDE100 and j2020 algorithms. The proposed algorithm j21 applies a population size reduction during the evolutionary process, which was not used in the two previous algorithms. It also uses an extended range of values for the self-adaptive control parameter CR.

The main contributions of this paper are: New algorithm (called j21) is proposed, computer experiments for CEC 2021 competition on the single objective real parameter optimization problems are conducted, the obtained results are collected and shown in a form required by the competition organizers, and we compare the obtained experimental results with some results of the classic DE algorithm and the j2020 algorithm – our algorithm from last year competition.

The remainder of the paper is as follows. An overview of

The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0041, and P2-0069).

978-1-7281-8393-0/21/$31.00 ©2021 European Union

Require: **P**... the population of *NP* individuals

Initialize population $\mathbf{P} = (\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_{NP})$ randomly
Initialize $F_i = 0.5$; $CR_i = 0.9$; $(i \in \{1, NP\})$
while stopping criteria is not met **do**
  for $(i \leftarrow 0;\ i < NP;\ i{+}{+})$ **do**
    ▷ *** jDE mutation ***
    $F_{tmp} \leftarrow \begin{cases} F_l + rand_1 * F_u, & \text{if } rand_2 < \tau_1, \\ F_{i,g}, & \text{otherwise,} \end{cases}$
    Randomly select r_1, r_2, and r_3; ▷ $r_1 \neq r_2 \neq r_3 \neq i$
    $\vec{v}_{i,g+1} \leftarrow \vec{x}_{r_1} + F_{tmp}(\vec{x}_{r_2} - \vec{x}_{r_3})$ ▷ DE/rand/1/
    ▷ *** jDE crossover ***
    $CR_{tmp} \leftarrow \begin{cases} rand_3, & \text{if } rand_4 < \tau_2, \\ CR_{i,g}, & \text{otherwise,} \end{cases}$
    $j_{rand} \leftarrow rand\{1, D\}$
    for $(j \leftarrow 0;\ j < D;\ j{+}{+})$ **do**
      if $(rand(0,1) \leq CR_{tmp}$ **or** $j = j_{rand})$ **then**
        $u_{i,j,g+1} \leftarrow v_{i,j,g+1}$
      else
        $u_{i,j,g+1} \leftarrow x_{i,j,g}$
      end if
    end for
    ▷ *** selection ***
    if $(f(\vec{u}_{i,g+1}) \leq f(\vec{x}_{i,g}))$ **then** ▷ minimization scenario
      $\vec{x}_{i,g+1} \leftarrow \vec{u}_{i,g+1}$
      $F_{i,g+1} \leftarrow F_{tmp}$
      $CR_{i,g+1} \leftarrow CR_{tmp}$
    else
      $\vec{x}_{i,g+1} \leftarrow \vec{x}_{i,g}$
      $F_{i,g+1} \leftarrow F_{i,g}$
      $CR_{i,g+1} \leftarrow CR_{i,g}$
    end if
  end for
end while

Ensure: $\vec{x}_{best}$ ... the best individual/vector
Ensure: $f(\vec{x}_{best})$ ... the function value of the best vector

Algorithm 1: jDE [16].

related work is presented in Section II. Section III describes our new algorithm, called j21. Experimental results of the j21 algorithm on the CEC 2021 benchmark function suite are presented in Section IV. The conclusion is given in Section V.

II. Background

In this section, we give a short background on DE and its two variants jDE and j2020, whose mechanisms are incorporated in our new algorithm.

A. Differential Evolution

Differential evolution (DE) [2] is a population-based evolutionary algorithm that achieves highly competitive performances in many real-world applications dealing with optimization [17], [18], [19], [20].

In the classic DE algorithm, a population **P** has NP vectors (individuals):

$$\mathbf{P}_g = (\vec{x}_{1,g}, \ldots, \vec{x}_{i,g}, \ldots, \vec{x}_{NP,g}),\ i = 1, 2, \ldots, NP.$$

g is employed as a generation index. Each individual:

$$\vec{x}_{i,g} = \{x_{i,1,g}, x_{i,2,g}, \ldots, x_{i,D,g}\}$$

is represented with D variables.

DE initializes population **P** and then evolves it throughout generations guiding the individuals in the searching process towards a global optimum. At the end of the optimization process, DE returns the best-found individual and its objective value.

Initialization: In this step, population **P** is generated in a way that each individual is initialized with randomly uniformly distributed values between lower and upper bound for all its variables:

$$x_{i,j,0} = x_{j,low} + rand_{i,j} \cdot (x_{j,upp} - x_{j,low}).$$

DE evolves the population and during each generation, DE employs the mutation, crossover, and selection operations for each individual. These operations are briefly described in the next subsections.

Mutation: A mutation is the first operation that is employed by DE. During the mutation operation, DE creates a mutant vector $\vec{v}_{i,g+1}$. Nowadays, many strategies exist how to create a mutant vector. Very often used in practice is the 'DE/rand/1' mutation strategy that has been introduced in the original DE algorithm [2]. This strategy randomly selects two vectors in a population and calculates their difference, then multiplies it by a scale factor F and adds to the third randomly selected vector. This mutation strategy can be written as follows [2]:

$$\vec{v}_{i,g+1} = \vec{x}_{r_1,g} + F \cdot (\vec{x}_{r_2,g} - \vec{x}_{r_3,g}).$$

where r_1, r_2, and r_3 are indices within a set of $\{1, ..., NP\}$ and they are randomly chosen in such a way that they are different from index i and pairwise different of each other:

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

There exist many other DE mutation strategies [3], [21], [22] and they have a different influence on a convergence speed on the one hand, and a different ability not being trapped in local optima during the evolutionary process on the other hand.

Crossover: The binomial crossover creates a trial vector $\vec{u}_{i,g+1}$ as follows:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1}, & \text{if } rand_{i,j} \leq CR \text{ or } j = j_{rand}, \\ x_{i,j,g}, & \text{otherwise,} \end{cases}$$

for $i = 1, 2, ..., NP$ and $j = 1, 2, ..., D$. CR denotes a crossover parameter, which is within the interval $(0, 1]$. It gives the probability of creating trial vector components from a mutant vector. In case that the component is not selected from the mutant vector, it is captured from the parent vector $\vec{x}_{i,g}$. The trial vector contains at least one component from the mutant vector. Randomly chosen index $j_{rand} \in \{1, 2, ..., NP\}$ takes care of that.

After crossover, if some trial vector's variables jump out of bounds of a search space, a restart or a repair mechanism is applied.

Besides the binomial crossover, another type of crossover used in DE is exponential crossover [3], [2].

Require: $\mathbf{P_b}$ ... large population
Require: $\mathbf{P_s}$ ... small population
Require: *bNP* ... size of $\mathbf{P_b}$
Require: *sNP* ... size of $\mathbf{P_s}$
Require: $bNP \geq sNP$ **and** $bNP = m \times sNP, m \in 1, 2, ...$

Initialize population $\mathbf{P_b} = (\vec{x}_1, \ldots, \vec{x}_{bNP})$ randomly
Initialize population $\mathbf{P_s} = (\vec{x}_1, \ldots, \vec{x}_{sNP})$ randomly
Initialize $F_i = 0.5$; $CR_i = 0.9$; $(i \in \{1, bNP{+}sNP\})$
while stopping criteria is not met **do**
  Check for a reinitialization of $\mathbf{P_b}$
  Check for a reinitialization of $\mathbf{P_s}$
  for each $i \in \mathbf{P_b}$ **do**
    ▷ *** on the large population ***
    perform jDE mutation (Steps 5–7 in Alg. 1) **using**
      $\vec{x}_{r_1} \in \mathbf{P_b}$
      $\vec{x}_{r_2}, \vec{x}_{r_3} \in \mathbf{P_b} \bigcup \mathbf{M_s}, \mathbf{M_s} \subset \mathbf{P_s}$
    end perform
    Perform **jDE crossover** (Steps 8–16 in Alg. 1)
    Perform crowding in $\mathbf{P_b}$
    Perform **jDE selection** (Steps 17–25 in Alg. 1)
  end for
  if $\vec{x}_{best} \in \mathbf{P_b}$ **then**
    Copy $\vec{x}_{best}$ into $\mathbf{P_s}$
  end if
  for $k \in 1, 2, ..., m$ **do** ▷ repeat m-times
    for each $i \in \mathbf{P_s}$ **do**
      ▷ *** on the small population ***
      Perform **jDE mutation** (Steps 5–7 in Alg. 1)
      Perform **jDE crossover** (Steps 8–16 in Alg. 1)
      Perform **jDE selection** (Steps 17–25 in Alg. 1)
    end for
  end for
  Check for a population size reduction of $\mathbf{P_b}$
end while

Ensure: $\vec{x}_{best}$ ▷ the best individual/vector
Ensure: $f(\vec{x}_{best})$ ▷ the value of the best vector

Algorithm 2: The proposed j21 algorithm. It is based on the self-adaptive differential evolution jDE100 [12] and j2020 [13] algorithms.

Selection: The selection operation compares the objective function values of two vectors, the population vector $\vec{x}_{i,g}$ and its corresponding trial vector $\vec{u}_{i,g+1}$. The better one will survive as a member of the next generation. Mathematically, this operation is in a case of minimization defined as follows:

$$\vec{x}_{i,g+1} = \begin{cases} \vec{u}_{i,g+1}, & \text{if } f(\vec{u}_{i,g+1}) \leq f(\vec{x}_{i,g}), \\ \vec{x}_{i,g}, & \text{otherwise.} \end{cases}$$

Stopping condition: The stopping criteria in DE can be defined as the maximum number of function evaluations ($MaxFEs$), the maximum number of generations (G_{MAX}), available execution time budget, etc.

B. jDE Algorithm

The jDE [16] algorithm applies the self-adapting mechanism of two control parameters (scale factor F and crossover rate CR) at the individual level. Before the mutation operation starts, new $F_{i,g+1}$ and $CR_{i,g+1}$ are generated as follows [16]:

$$F_{i,g+1} = \begin{cases} F_l + rand_1 * F_u, & \text{if } rand_2 < \tau_1, \\ F_{i,g}, & \text{otherwise,} \end{cases}$$

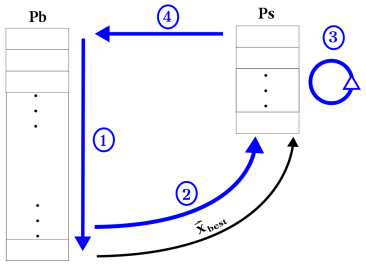

Fig. 1. Illustration of the evolutionary process in the jDE100 [12] and j2020 [13] algorithms. The algorithm repeats four steps: (1) Performs one generation on the larger population $\mathbf{P_b}$. (2) Copies the best individual ($\vec{x}_{best}$) into the smaller population $\mathbf{P_s}$. (3) Executes more generations on $\mathbf{P_s}$. (4) Switches the evolutionary process control back to $\mathbf{P_b}$.

TABLE I
SUMMARY OF THE CEC 2021 BOUND-CONSTRAINED REAL PARAMETER BENCHMARK FUNCTIONS [23].

	No.	Functions
Unimodal Functions	F1	Shifted and Rotated Bent Cigar Function (CEC 2017 F1)
Basic Functions	F2	Shifted and Rotated Schwefel's Function (CEC 2014 F11)
	F3	Shifted and Rotated Lunacek bi-Rastrigin Function (CEC 2017 F7)
	F4	Expanded Rosenbrock's plus Griewangk's Function (CEC2017 $f19$)
Hybrid Functions	F5	Hybrid Function 1 (N = 3) (CEC 2014 F17)
	F6	Hybrid Function 2 (N = 4) (CEC 2017 F16)
	F7	Hybrid Function 3 (N = 5) (CEC 2014 F21)
Composition Functions	F8	Composition Function 1 (N = 3) (CEC 2017 F22)
	F9	Composition Function 2 (N = 4) (CEC 2017 F24)
	F10	Composition Function 3 (N = 5) (CEC 2017 F25)
Search range: $[-100,100]^D$, $D = 10$, and D = 20		

$$CR_{i,g+1} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2, \\ CR_{i,g}, & \text{otherwise.} \end{cases}$$

Here, $rand_j$, $(j \in \{1, 2, 3, 4\})$ are random values that are uniformly distributed within the range $[0, 1]$. Values τ_1 and τ_2 are responsible to adjust control parameters F and CR, respectively. A pseudo-code of the jDE algorithm is shown in Alg. 1.

C. Algorithms jDE100 and j2020

In this section, we give an overview of two algorithms jDE100 [12] and j2020 [13]. The second one was presented in the previous year at the CEC conference, and it ranked in 3rd place, while the first one was presented at the CEC conference two years ago and ranked in the first place. Both algorithms are suitable for solving single-objective real-parameter optimization problems. They use two not equal-sized populations, i.e., a big population $\mathbf{P_b}$, and a small population $\mathbf{P_s}$. The evolutionary process over these two populations is shown in Figure 1. The population sizes in jDE100 are $bNP = 1000$ and $sNP = 25$, respectively, while in j2020 they are $7D$ and D.

TABLE II
PARAMETER SETTINGS IN THE PROPOSED J21 ALGORITHM.

Parameter	Value	Description
bNP	160	size of $\mathbf{P_b}$
sNP	10	size of $\mathbf{P_s}$
$F_{l,b}$	0.1	lower limit of scale factor for $\mathbf{P_b}$
$F_{l,s}$	0.17	lower limit of scale factor for $\mathbf{P_s}$
F_u	1.1	upper limit of scale factor for $\mathbf{P_b}$ and $\mathbf{P_s}$
$CR_{l,b}$	0.0	lower limit of crossover parameter for $\mathbf{P_b}$
$CR_{l,s}$	0.1	lower limit of crossover parameter for $\mathbf{P_s}$
$CR_{u,b}$	1.1	upper limit of crossover parameter for $\mathbf{P_b}$
$CR_{u,s}$	0.8	upper limit of crossover parameter for $\mathbf{P_s}$
F_{init}	0.5	initial value of scale factor
CR_{init}	0.9	initial value of crossover parameter
τ_1	0.1	probability to self-adapt scale factor
τ_2	0.1	probability to self-adapt crossover parameter
$ageLmt$	$\frac{maxFEs}{10}$	number of FEs with no improvement of the best individual then a restart in $\mathbf{P_b}$ is required to occurs
eps	10^{-12}	small value used to check if two function values are similar
$myEqs$	25	reinitialization if $myEqs\%$ of individuals in the corresponding population have the similar function values

TABLE III
RESULTS FOR 10D (BASIC).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	10.8737	10.8737	5.6323	5.4338
4	0.0000	0.4673	0.2574	0.2430	0.0997
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.2328	0.0194	0.0340	0.0662
7	0.0000	0.0377	0.0042	0.0080	0.0092
8	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	0.0036	47.9958	47.9880	46.3889	8.7608

TABLE IV
RESULTS FOR 10D (SHIFT OPERATOR).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0625	0.0000	0.0021	0.0114
3	0.0000	10.8737	10.8737	10.1488	2.7587
4	0.0319	0.4086	0.2655	0.2541	0.0797
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.2082	0.0195	0.0254	0.0397
7	0.0000	0.0357	0.0030	0.0056	0.0082
8	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0000	300.0000	100.0000	109.7698	47.6372
10	100.0117	400.0000	400.0000	363.3358	96.4236

TABLE V
RESULTS FOR 10D (ROTATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.2498	65.9362	10.3073	16.1707	17.0992
3	1.6251	16.1424	12.3438	11.5948	3.2553
4	0.3632	1.5393	0.7604	0.8443	0.2824
5	0.0000	12.9655	1.2035	4.3465	5.2843
6	0.0769	11.9144	0.5916	1.3233	2.8770
7	0.0061	0.6405	0.3278	0.3525	0.2141
8	0.0000	48.1326	0.0000	15.8585	19.8712
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	51.3506	52.2214	51.5835	51.6056	0.1781

The source code of both algorithms is available at https://github.com/P-N-Suganthan.

III. THE PROPOSED ALGORITHM J21

In this section, we present a new algorithm (called j21) for solving single-objective bound-constrained optimization problems. The pseudo-code of the j21 algorithm is given in Algorithm 2. The algorithm consists of several mechanisms, which are described in this section.

It uses self-adaptive F and CR control parameters, incorporated at the individual's level. This self-adaptation is the same as in the j2020 [13] algorithm, and it was used for the first time in the jDE [16] algorithm. If the trial vector has some variables out of bounds after crossover, then they are reflected into the search space. This is similar as in our previous algorithms.

The new algorithm applies two populations, which have different sizes. We denote a big population as $\mathbf{P_b}$ and a small population as $\mathbf{P_s}$. A similar approach of using two populations has been applied in the jDE100 [12] and j2020 [13] algorithms. The evolutionary process that is conducted over these two populations, is shown in Figure 1. The bigger population has bNP individuals and the smaller population has sNP individuals. Algorithm 2 starts with the initialization of both populations in Steps 1 and 2 and both self-adaptive control parameters F_i and CR_i for both populations. The evolutionary process is iterated in the main loop until a stopping criterion is met. The stopping criterion in this special session is set up with a predefined limit of the number of function evaluations, $maxFEs$.

The next two steps check if any population needs to be reinitialized. These two steps are similar in the j2020 algorithm.

The individuals in the big population $\mathbf{P_b}$ are reinitialized randomly if one of the following conditions is fulfilled: (1) $myEqs$ percent of the best individuals in $\mathbf{P_b}$ has converged (a difference of their function values is less than a small value $eps = 10^{-16}$), and (2) the best individual in the big $\mathbf{P_b}$ was not improved for $ageLmt$ evaluations.

Reinitialization in $\mathbf{P_s}$ occurs if $myEqs\%$ of sNP best individuals in $\mathbf{P_s}$ have converged (a difference of their function values is less than $eps = 10^{-16}$). All individuals in $\mathbf{P_s}$ are reinitialized, except the $\vec{x}_{best}$ individual in $\mathbf{P_s}$, which remains unchanged. One can see this mechanism with the best individual as elitism.

After that, the algorithm performs one generation on the big population (Lines 7–15 in Alg. 2). The well-known jDE/rand/1 mutation strategy is applied to generate a mutant vector $\vec{v}_i$. Note, r_1 indicates an index of $\mathbf{P_b}$ individual, while r_2 and r_3 indicate two individuals from $\mathbf{P_b} \cup \mathbf{M_s}$. Here, $\mathbf{M_s} \subset \mathbf{P_s}$ presents a small set of $\mathbf{P_s}$ individuals. Here, we want only the small influence from $\mathbf{P_s}$ on $\mathbf{P_b}$. There only indices r_2 and r_3 are used during this process but not r_1, which would have a bigger influence. The number of individuals in $\mathbf{M_s}$ in our

TABLE VI
RESULTS FOR 10D (TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	10.8737	0.0000	4.7119	5.4804
4	0.0000	0.3614	0.2574	0.2393	0.0928
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.2507	0.0197	0.0327	0.0550
7	0.0000	0.3239	0.0047	0.0198	0.0584
8	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	0.0461	48.0572	47.9880	46.3925	8.7534

TABLE VII
RESULTS FOR 10D (SHIFT AND ROTATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.4372	171.3836	11.8920	21.5458	34.2861
3	5.5558	15.1060	11.7437	11.7348	2.0313
4	0.1187	1.7470	0.7843	0.7956	0.3662
5	0.0000	11.5892	0.6246	1.7284	3.3110
6	0.0318	0.8982	0.5927	0.5166	0.2610
7	0.0000	17.0968	0.3130	1.3650	4.2406
8	0.0000	39.9225	0.0010	9.3124	11.9931
9	0.0000	100.0000	100.0000	90.0000	30.5129
10	100.0221	397.7457	397.7429	318.3631	133.8874

TABLE VIII
RESULTS FOR 10D (SHIFT AND TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0625	0.0000	0.0021	0.0114
3	0.0000	10.8737	10.8737	9.8537	2.9363
4	0.0319	0.4086	0.2566	0.2492	0.0762
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.2160	0.0197	0.0338	0.0531
7	0.0000	0.0499	0.0026	0.0077	0.0120
8	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0000	300.0000	100.0000	103.7816	48.0884
10	100.0134	400.0000	400.0000	388.9938	54.8571

TABLE IX
RESULTS FOR 10D (ROTATION AND TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.2498	65.9362	10.3073	16.1707	17.0992
3	0.0000	16.1424	12.0922	11.2593	3.9858
4	0.3632	1.5393	0.7604	0.8391	0.2844
5	0.0000	12.2366	1.2031	4.3366	5.2309
6	0.0769	11.9144	0.5916	1.3303	2.8787
7	0.0009	0.6395	0.3278	0.3411	0.2360
8	0.0000	53.3877	0.0000	12.3731	18.2466
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	51.3020	52.2214	51.5897	51.6150	0.2383

algorithm is not fixed, and it is expressed as follows:

$$|\mathbf{M_s}| = \begin{cases} 1, & \textbf{if} \quad 1 \leq it \leq \frac{1}{3}MaxFEs, \\ 2, & \textbf{if} \ \frac{1}{3}MaxFEs < it \leq \frac{2}{3}MaxFEs, \\ 3, & \textbf{if} \ \frac{2}{3}MaxFEs < it \leq MaxFEs, \end{cases}$$

where $it = 1, 2, \cdots, MaxFEs$ is the iteration counter. During the evolutionary process, we increase the number of individuals in $\mathbf{M_s}$. It is expected that $\mathbf{P_s}$, which has fewer individuals, has a faster convergence speed, but it also has a more chance to get trapped into a local minimum.

Then the crossover, crowding, and selection operations follow on $\mathbf{P_b}$. Crossover and selection are identical as in jDE. After a crossover operation, the j21 algorithm applies a crowding mechanism based on the Euclidean distance. An individual, which is the closest one to the trial vector, competes against the trial individual during a selection operation and the better one survives for the next generation.

When one generation in the big population $\mathbf{P_b}$ has finished, our algorithm checks if the best individual, $\vec{x}_{best}$, was found and then copies it into the small population $\mathbf{P_s}$. In such a way, the smaller population will have one fresh individual in the search process. This part is the same as in the j2020 algorithm.

Then the mutation, crossover, and selection operations are repeated m-times on the smaller population. In such a way, both populations are equipped with the equal number of function evaluations. Here, one can assume that $bNP = m \times sNP, m \in 1, 2, ...$, but, in general, this is not required.

The parameters of our j21 algorithm are collected in Table II. Parameter settings are similar as in the previous jDE100, j2020 algorithms. There are only small changes of some lower and upper limits' values, which were obtained by hand tuning.

Recently, a population size reduction mechanism is often applied in evolutionary algorithms. The linear decreasing population size reduction is used in the L-SHADE [24] algorithm and its derivations, like [25], iL-SHADE [26], jSO [1], LSHADE-EpSin [27], etc. Another type of population size reduction was used in works [28], [29], where a population is divided into two equal parts and an individual from the first part is competing with an individual from the second part and the better one is placed into a new population. The size of the new population is half of the population size before the reduction mechanism. Such a reduction mechanism can be applied several times during an evolutionary process and every time the population size is set to half.

In this paper, we used the population size reduction on the bigger population. The reduction mechanism is applied three times during the evolutionary process when the iteration counter it reaches:

- $\frac{1}{4}MaxFEs$,
- $\frac{2}{4}MaxFEs$,
- $\frac{3}{4}MaxFEs$.

In this case, the bigger population has four different sizes during the evolutionary process. For example, suppose that population size is 400 at the beginning of the evolutionary process, then within three population size reductions, the population size is reduced to 200, 100, and 50.

An evolutionary algorithm needs to apply a population diversity at a sufficiently high level in the early stages of the evolutionary process, while in the later stages the algorithm needs to reduce the population diversity to increase the convergence speed [30]. The population size reduction mechanism

TABLE X
RESULTS FOR 10D (SHIFT, ROTATION AND TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.4372	171.3836	11.8920	21.5458	34.2861
3	0.9950	14.8264	11.9393	11.4914	2.5345
4	0.1187	1.7470	0.7843	0.7951	0.3668
5	0.0000	11.5892	0.6246	1.7036	3.3167
6	0.0318	0.8982	0.5927	0.5166	0.2610
7	0.0000	17.0968	0.3130	1.9143	5.1150
8	0.0000	34.9887	11.5631	11.4069	10.4676
9	0.0000	100.0000	100.0000	90.0000	30.5129
10	100.0127	397.7457	397.7429	338.2016	121.1185

TABLE XI
RESULTS FOR 20D (BASIC).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	20.1617	20.1617	12.7691	9.8819
4	0.0000	0.5511	0.3752	0.3380	0.1369
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0073	0.2430	0.0707	0.0896	0.0621
7	0.0025	0.1826	0.0211	0.0353	0.0495
8	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	48.7532	48.7533	48.7532	48.7532	0.0000

TABLE XII
RESULTS FOR 20D (SHIFT OPERATOR).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	20.1617	20.1617	19.4897	3.6810
4	0.0000	0.4921	0.3315	0.3135	0.1267
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0055	0.4113	0.0626	0.0948	0.0983
7	0.0030	0.1890	0.0265	0.0324	0.0333
8	0.0000	100.0000	100.0000	90.0000	30.5129
9	100.0000	300.0000	300.0000	246.6667	89.9553
10	400.0000	400.0000	400.0000	400.0000	0.0000

TABLE XIII
RESULTS FOR 20D (ROTATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0937	5.4656	0.2186	0.8036	1.1450
3	0.0000	21.1844	20.3872	13.0716	9.9780
4	0.6518	1.1877	0.9031	0.9065	0.1339
5	1.3706	136.6971	15.3632	28.4332	35.9765
6	0.1681	3.5406	0.6243	0.7701	0.6186
7	0.1522	63.0947	9.0615	14.9399	16.3903
8	0.0000	126.1177	96.7846	92.6132	25.2404
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	62.5164	63.1648	62.8854	62.8672	0.1711

is a step toward providing a proper population diversity level during the evolutionary process.

The last mechanism used in our j21 algorithm is an extended range of values for the self-adaptive control parameter CR, which was introduced for solving large-scale global optimization problems in algorithm LSGOjDE [22]. It is controlled by the $CR_{u,b}$ parameter and is applied only in the bigger population in the j21 algorithm. A value of the crossover parameter CR is usually less or equal to 1 but in our algorithm, we proposed a value that is greater than 1. Consequently, when crossover parameter $CR > 1$, the rotational invariant strategy takes place. This strategy applies only mutation but no crossover operation.

Notice, the population update of jDE (and its variants, including j21) differs from the normal DE population update (asynchronous vs synchronous update) wherein every newly generated individual is added to the population and can take part in the mutation of later individuals. This somehow helps jDE to maintain the selection pressure instead of using a variant of the DE/rand/1 update.

IV. EXPERIMENTS

A. Benchmark Functions

The benchmark functions for the CEC 2021 single objective bound-constrained optimization are presented [23]. In this competition, the benchmark functions are parameterized by the operators such as bias, rotation, and shift. The main motive behind the parameterization is to test the effect of all combinations of the operator on all benchmark functions.

Table I shows the CEC 2021 benchmark functions suite. It contains one unimodal benchmark function, three basic functions, three hybrid functions, and three composition functions. A search space is determined by the upper and lower bounds and they are defined as $[-100, 100]^D$. The benchmark functions have dimensions $D = 10$ and 20, and all functions are scalable. The goal is to perform a minimization for each benchmark function. This special session competition, as several previous competitions, except CEC 2019, had preset the limitation of the maximum number of function evaluations $maxFEs$. An algorithm's run needs to terminate after $maxFEs$ function evaluations. For all benchmark functions, $maxFEs$ is defined as follows:

- $maxFEs$ =200,000 for $D = 10$,
- $maxFEs$ =1,000,000 for $D = 20$.

The both values of the maximum number of function evaluations for this year are set lower than those at the CEC 2020 competition, where $maxFEs$ =1,000,000 for $D = 10$, and $maxFEs$ =10,000,000 for $D = 20$.

There are eight different transformations for each function. The optimal solution values are known for all benchmark functions for each transformation (see Table 3 in [23]).

Next, it is required to record the function value ($F_i^* = F_i(\vec{x}^*)$) after $\left\lfloor D^{\frac{k}{5}-3} maxFEs \right\rfloor$ ($k = 0, 1, 2, 3, ..., 15$) for each configuration for each run. Therefore, 16 error values are recorded for each benchmark function for each run and repeated for each configuration. The obtained final results in the specified format are asked by the organizers who will prepare an overall analysis and comparison based on these results of all contestants. Note, error values that are smaller than 10^{-8} are considered as zero.

In this competition, 30 consecutive runs of an algorithm are needed for each function for each of the 8 configurations,

TABLE XIV
RESULTS FOR 20D (TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	20.1617	20.1617	10.0809	10.2532
4	0.0198	0.5511	0.3646	0.3337	0.1318
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0042	0.2276	0.0659	0.0832	0.0650
7	0.0017	0.1821	0.0238	0.0359	0.0500
8	0.0000	0.0000	0.0000	0.0000	0.0000
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	48.7532	48.7533	48.7532	48.7532	0.0000

TABLE XV
RESULTS FOR 20D (SHIFT AND ROTATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0625	3.4531	0.2498	0.6489	0.8674
3	20.3872	21.2252	20.3872	20.4962	0.1876
4	0.5500	1.3001	0.8576	0.8824	0.1805
5	4.6668	222.6899	46.2997	71.3088	66.1034
6	0.1123	0.5697	0.3187	0.3063	0.1226
7	0.2951	128.0604	17.2467	28.0297	34.9147
8	53.9198	100.0000	100.0000	98.4640	8.4131
9	100.0000	418.7540	408.0673	310.5969	137.9921
10	399.0652	410.1458	399.2991	400.8697	3.7321

TABLE XVI
RESULTS FOR 20D (SHIFT AND TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	20.1617	20.1617	19.4897	3.6810
4	0.0000	0.4921	0.3371	0.3206	0.1284
5	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0063	0.2948	0.0605	0.0856	0.0799
7	0.0005	0.2215	0.0265	0.0359	0.0444
8	0.0000	100.0000	100.0000	93.3333	25.3708
9	100.0000	300.0000	300.0000	286.6667	50.7416
10	400.0000	400.0000	400.0000	400.0000	0.0000

TABLE XVII
RESULTS FOR 20D (ROTATION AND TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0937	5.4656	0.2186	0.8036	1.1450
3	0.0000	20.8820	20.3872	12.3441	10.0974
4	0.6518	1.1877	0.9031	0.9065	0.1339
5	1.3706	136.6971	15.3632	28.4372	35.9749
6	0.1671	3.5121	0.5380	0.7522	0.6064
7	0.1522	63.0947	9.0615	14.9399	16.3903
8	65.6498	125.1987	93.5111	95.3102	16.2681
9	0.0000	0.0000	0.0000	0.0000	0.0000
10	62.5630	63.2270	62.8147	62.8543	0.1973

and the error values achieved after the maximum number of function evaluations $maxFEs$ are collected and analyzed. The best, worst, mean, median, and standard deviation values of the function error values for the 8 configurations and 30 runs are presented in the next sub-section.

B. Experimental Results

The obtained results in the experiments of the proposed j21 algorithm are shown in Tables III–XVIII. There are 16 tables, one for each dimension and each configuration (there are 8 configurations for dimension $D = 10$, and 8 configurations for $D = 20$).

A comparison based on the mean values on $D = 20$ (shift, rotation, and translation) of the original DE, j2020 and the proposed j21 algorithms is shown in Table XX. The best-obtained results for each function in this configuration of the compared algorithms are shown in **bold**. The original DE used the following parameter settings: $F = 0.5$, $CR = 0.9$, and $NP = 100$ for all functions, configurations and dimensions. The parameter settings for j2020 are as described in [13]. One can notice a similar performance of all algorithms on function F1. The proposed j21 algorithm obtained the best results on functions F2–F4, F6, F9, and F10, while the original DE algorithm obtained the best results on functions F5 and F7.

Since the compiler on our system has made a simplification during the optimization phase of compilation:

```
x=x+x; x=x/2; x=x*x; x=sqrt(x);
x=log(x); x=exp(x); y=x/x;
```

which prevents us to compute $T0$ (it was zero), we slightly change the code:

```
x=x+x+0.0001; x=x/2; x=x*x+0.0001; x=sqrt(x);
x=log(x)+0.0001; x=exp(x); y=(x+0.0001)/x;
```

to compute/measure $T0$. The measured execution times are shown in Table XIX.

In our experimental work we used PC with the next configuration: System: GNU Linux, CPU: Intel(R) Core(TM) i7-4770 CPU 3.4 GHz, Main memory: 16 GB, Programming language: C++, Algorithm: j21, Compiler: g++.

V. CONCLUSIONS

In this paper, we presented a new differential evolution algorithm for solving single-objective real-parameter bound-constrained optimization problems. The proposed algorithm uses several mechanisms to tackle optimization problems efficiently: two populations with different sizes, restart mechanism in both populations, self-adaptive control parameters F and CR, the extended range of values for CR in the bigger population, migration of the best individual from the big population into the small population, modified mutation strategy in the bigger population, crowding mechanism and population size reduction in the bigger population.

Ten benchmark functions with eight different configurations for dimensions $D = 10$ and $D = 20$ have been used in experimental work. The obtained results of the proposed algorithm j21 are collected in tables as required by the organizers of the CEC 2021 competition.

Additionally, we compared the results of our j21 algorithm with the original DE and j2020 algorithms on problems with dimension $D = 20$ on one selected configuration (shift, rotation, and translation). The proposed j21 algorithm shows better overall performance.

TABLE XVIII
RESULTS FOR 20D (SHIFT, ROTATION AND TRANSLATION).

Func.	Best	Worst	Median	Mean	Std
1	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0625	3.4531	0.2498	0.6489	0.8674
3	20.3872	21.3585	20.3872	20.5172	0.2438
4	0.5500	1.3001	0.8576	0.8824	0.1805
5	4.6668	222.6899	42.6402	67.4726	66.1401
6	0.0729	0.5697	0.2820	0.3065	0.1394
7	0.2951	128.0604	17.2467	28.0297	34.9147
8	36.1164	100.0000	100.0000	96.0504	15.0801
9	100.0000	423.3446	412.2307	330.1160	133.5977
10	399.0080	410.1154	399.1706	400.4787	3.3160

TABLE XIX
COMPUTATION COMPLEXITY.

	T0	T1	*T2*	*(T2 - T1)/T0*
D =10	1e-06	0.01297	0.3551	3.421e+05
D =20	1e-06	0.02457	0.7864	7.618e+05

TABLE XX
COMPARISON OF THE MEAN RESULTS FOR 20D (SHIFT, ROTATION AND TRANSLATION).

Func.	DE	j2020	j21
1	**0.0000**	**0.0000**	**0.0000**
2	859.3452	2.4272	**0.6489**
3	27.6133	20.5774	**20.5172**
4	3.5780	1.2207	**0.8824**
5	**48.1135**	283.1423	67.4726
6	1.3610	0.4156	**0.3065**
7	**10.2144**	31.1123	28.0297
8	100.0000	100.0000	**96.0504**
9	407.4073	386.8001	**330.1160**
10	413.4250	403.7982	**400.4787**

REFERENCES

[1] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: Algorithm jSO," in *IEEE Congress on Evolutionary Computation (CEC) 2017*. IEEE, 2017, pp. 1311–1318.

[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. M. Ali, D. Essam, and K. Kasmarik, "A novel design of differential evolution for solving discrete traveling salesman problems," *Swarm and Evolutionary Computation*, vol. 52, p. 100607, 2020.

[6] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[7] T. Eltaeib and A. Mahmood, "Differential Evolution: A Survey and Analysis," *Applied Sciences*, vol. 8, no. 10, p. 1945, 2018.

[8] P. Sharma, H. Sharma, S. Kumar, and J. C. Bansal, "A Review on Scale Factor Strategies in Differential Evolution Algorithm," in *Soft Computing for Problem Solving*. Springer, 2019, pp. 925–943.

[9] M. S. Maučec and J. Brest, "A review of the recent use of Differential Evolution for Large-Scale Global Optimization: An analysis of selected algorithms on the CEC 2013 LSGO benchmark suite," *Swarm and Evolutionary Computation*, vol. 50, p. 100428, 2019.

[10] K. R. Opara and J. Arabas, "Differential Evolution: A survey of theoretical analyses," *Swarm and evolutionary computation*, vol. 44, pp. 546–558, 2019.

[11] K. V. Price, N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization," Nanyang Technological University, Singapore, Tech. Rep., November 2018. [Online]. Available: http://www.ntu.edu.sg/home/epnsugan/

[12] J. Brest, M. S. Maučec, and B. Bošković, "The 100-Digit Challenge: Algorithm jDE100," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 19–26.

[13] ——, "Differential evolution algorithm for single objective bound-constrained optimization: Algorithm j2020," in *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020, pp. 1–8.

[14] K. V. Price, N. H. Awad, M. Z. Ali, and P. N. Suganthan, "The 2019 100-Digit Challenge on Real-Parameter, Single Objective Optimization: Analysis of Results." [Online]. Available: https://github.com/P-N-Suganthan/CEC2019

[15] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali., B. Y. Qu, N. H. Awad, and P. P. Biswas, "Competition on Single Objective Bound Constrained Numerical Optimization." [Online]. Available: https://github.com/P-N-Suganthan/CEC2020

[16] 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.

[17] B. Bošković and J. Brest, "Two-phase protein folding optimization on a three-dimensional ab off-lattice model," *Swarm and Evolutionary Computation*, vol. 57, p. 100708, 2020.

[18] P. Krömer, J. Platoš, and V. Snášel, "Differential evolution for the optimization of low-discrepancy generalized halton sequences," *Swarm and Evolutionary Computation*, vol. 54, p. 100649, 2020.

[19] Z. Liao, W. Gong, L. Wang, X. Yan, and C. Hu, "A decomposition-based differential evolution with reinitialization for nonlinear equations systems," *Knowledge-Based Systems*, vol. 191, p. 105312, 2020.

[20] J. Dugonik, B. Bošković, J. Brest, and M. Sepesy Maučec, "Improving statistical machine translation quality using differential evolution," *Informatica*, vol. 30, no. 4, pp. 629–645, 2019.

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

[22] M. S. Maučec, J. Brest, B. Bošković, and Z. Kačič, "Improved Differential Evolution for Large-Scale Black-Box Optimization," *IEEE Access*, 2018.

[23] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, P. Agrawal, A. Kumar, P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2021 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization," Nanyang Technological University, Singapore, Tech. Rep., November 2020. [Online]. Available: https://github.com/P-N-Suganthan/2021-SO-BCO

[24] R. Tanabe and A. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," in *2014 IEEE Congress on Evolutionary Computation (CEC2014)*. IEEE, 2014, pp. 1658–1665.

[25] K. Sallam, R. Sarker, D. Essam, and S. Elsayed, "Neurodynamic differential evolution algorithm and solving cec2015 competition problems," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, May 2015, pp. 1033–1040.

[26] J. Brest, M. S. Maučec, and B. Bošković, "iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization," in *IEEE Congress on Evolutionary Computation (CEC) 2016*. IEEE, 2016, pp. 1188–1195.

[27] N. H. Awad, M. Z. Ali, P. N. Suganthan, and R. G. Reynolds, "An ensemble sinusoidal parameter adaptation incorporated with L-SHADE for solving CEC2014 benchmark problems," in *IEEE Congress on Evolutionary Computation (CEC) 2016*. IEEE, 2016, pp. 2958–2965.

[28] J. Brest and M. S. Maučec, "Population Size Reduction for the Differential Evolution Algorithm," *Applied Intelligence*, vol. 29, no. 3, pp. 228–247, 2008.

[29] 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*, vol. 15, no. 11, pp. 2085–2087, 2011.

[30] 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.