2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Computing Opposition By Involving Entire Population

Shahryar Rahnamayan, *Senior Member, IEEE,* Jude Jesuthasan, Farid Bourennani,
Hojjat Salehinejad, *Student Member, IEEE,* and Greg F. Naterer

***Abstract*—The capabilities of evolutionary algorithms (EAs) in solving nonlinear and non-convex optimization problems are significant. Among the many types of methods, differential evolution (DE) is an effective population-based stochastic algorithm, which has emerged as very competitive. Since its inception in 1995, many variants of DE to improve the performance of its predecessor have been introduced. In this context, opposition-based differential evolution (ODE) established a novel concept in which, each individual must compete with its opposite in terms of the fitness value in order to make an entry in the next generation. The generation of opposite points is based on the population's current extreme points (i.e., maximum and minimum) in the search space; these extreme points are not proper representatives for whole population, compared to centroid point which is inclusive regarding all individuals in the population. This paper develops a new scheme that utilizes the centroid point of a population to calculate opposite individuals. Therefore, the classical scheme of an opposite point is modified accordingly. Incorporating this new scheme into ODE leads to an enhanced ODE that is identified as centroid opposition-based differential evolution (CODE). The performance of the CODE algorithm is comprehensively evaluated on well-known complex benchmark functions and compared with the performance of conventional DE, ODE, and some other state-of-the-art algorithms (such as SaDE, ADE, SDE, and jDE) in terms of solution accuracy. The results for CODE are promising.**

I. Introduction

DIFFERENTIAL evolution (DE), an evolutionary algorithm (EA) proposed by Storn and Price in 1995 [1], [2], presents a higher performance compared to other EAs because of its simplicity, effectiveness, and lower number of control parameters. Over the past decade, DE has been widely used to solve global optimization problems in various engineering and science fields. Similar to other EAs, DE is a population-based stochastic algorithm which can suffer from slow convergence speed depending on the complexity of the problem. Many researchers investigated how to accelerate DE; a comprehensive survey of DE-related state-of-the-art work can be found in [3]. Research works have undertaken to improve the performance of DE by control parameter fine-tuning and trial vector generation strategies for specific problems [4]-[7], as well as obtaining those values and strategies self-adaptively [8], [9]. Amidst all of the above mentioned developments, Rahnamayan et al. [10] proposed a new variant of DE known as Opposition-based Differential Evolution (ODE) to accelerate DE convergence speed and obtain more accurate solutions. By using opposition-based learning concept [11], the ODE benefits from a stronger exploration capability resulting in faster convergence. In ODE, a scheme known as opposition-based computation (OBC) was utilized to generate opposite of current candidate solutions. OBC defines an opposite-point based on a predefined relationship between the extreme points (i.e., max and min) of a current population and the trial point. Thus, applying OBC to a candidate-solution generated by DE means that the opposite solution is calculated by using coordinates of the candidate solution, the maximum and the minimum of the population along each dimension as boundary points. It has been shown that ODE outperforms DE [11]. ODEs performance can be further improved by having an opposition learning scheme that would account for all members of a population rather than accounting for only the minimum and maximum (extremes) of the population.

In this paper, a new OBC scheme, called centroid opposition-based computation (COBC), is proposed to meet the above features. Almost 200 OBL related papers are published so far, but only this paper considers the entire population in its opposition scheme, by using the center of gravity, rather than the min and max points. The centroid is the point where the centre of mass lies in a uniform body. For this matter, it is assumed that the entire population of DE is a discrete body; so the unit mass is distributed. The use of a centroid does not only strengthen the learning process of an algorithm, even with no boundaries, they are calculated at the first iteration. In this paper, a new variant of ODE algorithm that uses the COBC scheme is proposed and named centroid opposition-based different evolution (CODE). Experimental results have confirmed that CODE is much faster than both ODE and DE and it also provides promising results when compared with its other adaptive and self-adaptive versions of DE. The concept of OBC has been used in many well-known machine learning and optimization techniques such as ant colony system (ACS) [14] and genetic algorithm (GA) [33]. This idea can also be developed to increase of proposed methods for in wireless communication systems [25], and vehicular navigation systems [30].

The remaining of the paper is organized as follows. The conventional DE will be briefly explained in Section II. In Section III, a brief review of OBC is provided and the proposed COBC approach is described. The COBC scheme

Shahryar Rahnamayan, Farid Bourennani, and Hojjat Salehinejad are with the Department of Electrical, Computer, and Software Engineering, University of Ontario Institute of Technology, 2000 Simcoe Street North, Oshawa, ON L1H 7K4, Canada (emails: {shahryar.rahnamayan, farid.bourennaniand, hojjat.salehinejad}@uoit.ca).

Jude Jesuthasan is with the Electrical and Computer Engineering Department, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada (email: Jesuthasan@uwaterloo.ca)

Greg. F. Naterer is with the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, St. John's, Canada (email: dean.engineering@mun.ca).

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

is incorporated into DE to provide a new variant of DE called CODE in Section IV. The performance of CODE will be compared with DE, ODE, and other state-of-the-art algorithms in Section V. Finally, the paper is concluded and some further challenges are introduced in Section VI.

II. Differential Evolution

DE is an effective population-based optimization algorithm which can be utilized to solve a global optimization problem formalized as follows

$$Minimize\ f(\mathbf{X}), \mathbf{X} = [x_1, ..., x_D] \in R, \tag{1}$$

subject to

$$g_i(\mathbf{X}) \leq 0,\ for\ i = 1, ..., p, \tag{2}$$

where $x_i^L \leq x_i \leq x_i^U$ for $i = 1, ..., D$, D is the dimension of the problem, p is the number of constraints, and x_i^L and x_i^U indicate the lower and upper bounds of the variable x_i, respectively. DE operates through four stages, described below, in order to find the desired optimal solution.

A. Population Initialization

This initiates the search towards the global optimum by having NP stage number of individuals in population, D dimensional, uniform randomly generated candidate solutions, which are known as initial parameter vectors. Since each parameter vector is subject to change over a limited number of generations G, it is customary to denote the i^{th} parameter vector of G^{th} generation as follow:

$$\mathbf{X}_{i,G} = [x_{1,i,G}, ..., x_{D,i,G}], \tag{3}$$

where $G = 0, ..., G_{Max}$ and G_{Max} indicates the maximum number of the generations. During the initialization stage, a component of a parameter vector $x_{j,i,0}$ is initialized according to the following equation

$$x_{j,i,0} = x_j^L + rand_{i,j}(0,1) \times [x_j^U - x_j^L], \tag{4}$$

where $j = 1, .., D$, $i = 1, ..., NP$, $rand(0,1)$ generates a uniform random number in $[0,1]$, and x_i^L and x_i^U are the lower and upper bounds of the variable x_i, respectively.

B. Mutation Operator

A mutation operator generates a vector known as a donor vector $\mathbf{V}_{i,G}$ for each vector in the current population identified as a target vector. Although there are variant DE-mutation schemes [4], [5], the classical version is given as

$$\mathbf{V}_{i,G} = \mathbf{X}_{r1,G}^i + F(\mathbf{X}_{r2,G}^i - \mathbf{X}_{r3,G}^i), \tag{5}$$

where $i = [1, ..., NP]$. Here, the indices $r1$, $r2$, $r3$ are mutually exclusive integers chosen randomly from the set $\{1, ..., NP\}$. Furthermore, the value of the amplification factor F of the difference vector typically lies in the interval $(0, 2]$.

C. Crossover Operator

By shuffling a donor vector with its associated target vector to enhance the potential diversity of the population, this phase results in a vector known as a trial vector $\mathbf{U}_{i,G}$ defined as follows:

$$U_{j,i,G} = \begin{cases} v_{j,i,G}, & rand_{i,j}(0,1) \leq Cr\ or\ j = rand_j \\ x_{j,i,G}, & otherwise \end{cases}, \tag{6}$$

where $rand_{i,j}(0,1)$ is the j^{th} uniformly distributed random number generated for the i^{th} trial vector, $Cr \in (0,1)$ is a constant crossover rate, and $rand_j \in \{1, ..., D\}$ is a random integer number, where ensures $\mathbf{U}_{i,G}$ inherits at least one component from $\mathbf{V}_{i,G}$.

D. Selection

Finally, this step leads to a new generation $G + 1$, which is derived by having made the selection either to retain the old solution $x_{i,G}$ or introduce a new candidate solution $\mathbf{U}_{i,G}$ instead. For a minimization problem, it is defined as follows:

$$\mathbf{X}_{i,G+1} = \begin{cases} \mathbf{U}_{i,G}, & f(\mathbf{U}_{i,G}) \leq f(X_{i,G}) \\ \mathbf{X}_{i,G}, & f(\mathbf{U}_{i,G}) > f(\mathbf{X}_{i,G}) \end{cases}, \tag{7}$$

where $i = 1, ..., NP$. A comprehensive survey about DE can be found in [22].

III. Opposition-Based Learning

Opposition-based learning (OBL) was introduced by Tizhoosh in 2005 [12]. The main idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate (i.e., guess and opposite guess) in order to achieve a more accurate approximation for the current candidate solution. By considering opposite individuals during opposition-based population initialization and generation jumping, OBL was successfully applied to DE to solve well-known benchmark problems [13],[14], noisy problems [15], and large scale problems [16] more effectively. A self-adaptive ODE was also introduced in [17]. Another version of OBL used quasi-opposite numbers rather than opposite numbers in ODE to form the Quasi-Oppositional DE (QODE) [18], [19]. Both ODE and QODE used a constant generation jumping rate. The variable jumping rates were investigated for ODE in [20]. A lower jumping rate presented a better performance than a fixed rate, which means opposition-based generation jumping is more beneficial during exploration than exploitation. Among all proposed opposition-based algorithms, the ODE is the most well-known method. Weber et al. [21] divided modern DE-based algorithms into the following two categories: (1) DE with an integrated extra component, and (2) DE with a modified structure. The first group includes the algorithms with a DE framework and an extra component, such as local searchers and/or additional operators. The second group contains types of DE-based algorithms which modify the main structure of the canonical DE. According to this classification, the same authors considered the ODE in the second category [22] with other recently proposed enhanced DE variants, such as self adaptive control parameters [23], global-local search

DE [24], and self adaptive coordination of multiple mutation rules [25].

DE suffers from a limited amount of exploratory moves due to its limited mutation and crossover combinations which can be improved by embedding alternative moves [21]. Furthermore, the limited amount of moves can cause undesirable search process stagnation; a situation where diversity of the population is still high but it does not converge to a solution [26]. The successful extra moves can be achieved by two ways: (a) increasing the exploratory pressure, and/or (b) utilizing a randomization scheme [22]. In this regard, the ODE uses the first approach by proposing a new operator, i.e. opposition-based generation jumping which checks unexplored areas of the decision space by utilizing the above mentioned alternative moves [22]. These additional moves improve the DE exploration performance and also reduce the chance of stagnation by injecting fitter opposite individuals during the generation jumping. The risk of stagnation of DE is higher when the dimension of the problem increases [22], [23]. This may explain why in general, ODE performs better in large-scale problems [16]. Following, the traditional min-max OBC scheme is presented and then the proposed centroid-based OBC is introduced.

A. Min-Max Oppositional Computing

In a one-dimensional search space, let $x \in [a, b]$ where $[a, b] \subset R$. The opposite point $\check{x}$ of x is defined as follows:

$$\check{x} = a + b - x. \tag{8}$$

In a multidimensional space, let $P = [x_1, ..., x_D]$ be a point in D-multidimensional space with $x_i \in [a_i, b_i]$ for $i = 1, ..., D$. Then, the opposite point $\check{P} = [\check{x}_1, ..., \check{x}_D]$ is defined as

$$\check{x}_i = a_i + b_i - x_i, \tag{9}$$

for $i = 1, ..., D$. Upon the application of OBC to a candidate solution at a particular generation, its opposite is calculated according to Eq. (9). The best candidate solutions are selected based on their corresponding fitness value. For instance, assume that f is the given fitness function that must be minimized, and P and $\check{P}$ are the current population and corresponding opposite population, respectively. By comparing the fitness values of candidate solutions of $\{\check{P} \cup P\}$, the candidate solutions with better fitness values are selected to go through the evolutionary process while the other candidate solutions omitted. This process is called opposition-based optimization. The above approach was applied during opposition-based population initialization and generation jumping, which are described in the following section. These two components are embedded into DE to build ODE [10]. In this paper, the same two components are used but with a new centroid opposition-based scheme, which is described in the following subsection.

B. Min-Max vs. Centroid Oppositional Computing

In ODE, when an opposite candidate solution is calculated, it is necessary to select two boundary points for each dimension. This is performed by choosing the maximum and minimum values of a variable j from the population of vectors. ODE uses OBC to accelerate DE which results in efficient performance on average for a majority of benchmark problems [10]. However, still there is a room of further enhancement of OBC scheme. When calculating the opposite point, two boundaries (min and max) are taken from two extreme points in the population for every dimension and the remaining points of the population are not considered. The proposed COBC method takes the entire population for the generation of the opposite which improves the ODE in terms of convergence speed and solution accuracy. The centroid point and centroid-opposition based point are defined as below.

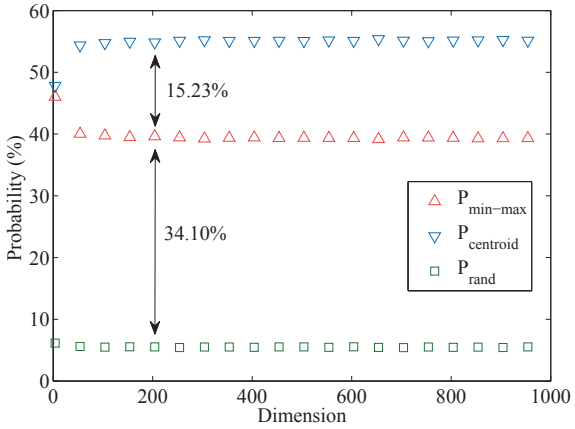

Fig. 1. Monte-Carlo simulation for the min-max, centroid, and rand oppositional-based computing (OBC) methods.

Let $(\mathbf{X}_1, ..., \mathbf{X}_n)$ be n points in D dimensional search space with each point in that space carrying a unit mass. Then the centroid of the body can be defined as follows:

$$\mathbf{M} = \frac{\mathbf{X}_1 + ... + \mathbf{X}_n}{n}, \tag{10}$$

where then we have

$$M_i = \frac{\sum_{j=1}^{n} x_{i,j}}{n}. \tag{11}$$

Having defined the centroid of a discrete uniform body as $\mathbf{M}$, the opposite-point $\check{\mathbf{X}}_i$ of a point $\mathbf{X}_i$ of the body is calculated as follows

$$\check{\mathbf{X}}_i = 2 \times \mathbf{M} - \mathbf{X}_i. \tag{12}$$

It is shown using Monte-Carlo method, without using minimum and maximum boundaries of landscape, the centroid approach can be employed with a better performance than min-max method. The simulation is conducted for dimensions one to one-thousand and with $NRun = 10,000$ number of independent runs per dimension. In each run, $NS = 50$ sample points as well as one point, as an optimum, are uniform randomly generated in interval $[X_{min}, X_{max}]$. Then, the estimated boundary based on the generated sample points is calculated as $[x_{min}, x_{max}]$. In each independent run

r for each dimension d, the Euclidean distance between the opposite of a sample point and the optimal point is calculated. Then, the minimum distance is selected as

$$\Delta^r = min\{\delta_1^r, ..., \delta_{NS}^r\}. \tag{13}$$

By considering the min-max, centroid, and random placement of opposite points (rand), approaches for comparison, the distances using both approaches are calculated and then cardinality (the number of runs) in which each one was successful is calculated for centroid method as

$$n_{centroid} = card\{\Delta_{centroid}^r < min\{\Delta_{min-max}^r, \Delta_{rand}^r\}\}, \tag{14}$$

for min-max method as

$$n_{min-max} = card\{\Delta_{min-max}^r < min\{\Delta_{centroid}^r, \Delta_{rand}^r\}\}, \tag{15}$$

and for rand method as

$$n_{rand} = card\{\Delta_{Rand}^r < min\{\Delta_{min-max}^r, \Delta_{centroid}^r\}\}. \tag{16}$$

By using the definitions in Eq.s (14)-(16), success probability of min-max, centroid, and rand schemes are illustrated in Fig. 1. By having in mind the total number of runs, the success probability for min-max, centroid, and rand schemes are defined as following, respectively:

$$P_{min-max} = \frac{n_{min-max}}{NRun}, \tag{17}$$

$$P_{centroid} = \frac{n_{centroid}}{NRun}, \tag{18}$$

and

$$P_{rand} = \frac{n_{rand}}{NRun}, \tag{19}$$

where $P_{centroid} + P_{min-max} + P_{Rand} = 1$. Therefore, for $d = 200$, $P_{centroid} = 54.85\%$, $P_{min-max} = 39.62\%$, and $P_{rand} = 5.52\%$ and for $d = 1000$, $P_{centroid} = 55.14\%$, $P_{min-max} = 39.34\%$, and $P_{rand} = 5.52\%$.

IV. Centroid Opposition-Based Differential Evolution

Similar to ODE, CODE has the same two main added opposition-based components, namely opposition-based population initialization and opposition-based generation jumping.

A. Opposition-Based Population Initialization

At this stage, after initialization of the population P of size NP with D dimensional vectors at generation zero, its opposite population $\check{P}$ is computed by using a COBC scheme. Unlike an opposition point n_c which is generated using an OBC scheme, COBC can generate opposite points outside the search space. This instance is avoided by randomly initializing the opposite-point in between the centroid point M, and the boundary point that lie in the same direction as the opposite point. By using the scheme of opposition-based optimization, NP fittest parameter vectors are selected from $\{P \cup \check{P}\}$ to go through the evolution process.

Algorithm 1 Centroid Opposition-based Differential Evolution (CODE)

Procedure CODE
// Opposition-based population initialization
Generate uniformly distributed random population P_0
// Centroid point evaluation
for $i = 1 \rightarrow D$ **do**
  $M_i = 0$
  for $j = 1 \rightarrow NP$ **do**
    $M_i = M_i + P_{0,i,j}$
  end for
end for
$\mathbf{M} = \mathbf{M}/NP$ ***// End of centroid point evaluation***
// Centroid opposite population calculation
for $i = 1 \rightarrow NP$ **do**
  for $j = 1 \rightarrow D$ **do**
    $OP_{0,i,j} = 2 \times M_j - P_{0,i,j}$
  end for
end for
// End of centroid opposite population calculation
Randomly initialize opposite points between M_i and the boundary point in the opposite direction if any opposite points are outside the search space.
Select NP fittest individuals from the set $\{P_0 \cup OP_0\}$ as initial population P_0
// End of opposition-based population initialization
while ($BFV > VTR$ & $NFC < NFC_{Max}$) **do**
  for $i = 1 \rightarrow NP$ **do**
    Evaluate the donor vector $V_{i,G}$
    Evaluate the trial vector $U_{i,G}$
    // Selection of parameter vectors of the next generation
    if $f(U_{i,G}) \leq f(P_{i,G})$ **then**
      $P_{i,G+1} = U_{i,G}$
    else
      $P_{i,G+1} = P_{i,G}$
    end if
    // End of selection of parameter vectors of the next generation
  end for
  // Opposition-based generation jumping
  Update the search space to $min_j P$, $max_j P$
  if $rand(0,1) < J_r$) **then**
  // Centroid point evaluation
    for $i = 1 \rightarrow D$ **do**
      $M_i = 0$
      for $j = 1 \rightarrow NP$ **do**
        $M_i = M_i + P_{G,i,j}$
      end for
    end for
    $\mathbf{M} = \mathbf{M}/NP$ ***// End of centroid point evaluation***
  // Centroid opposite population calculation
    for $i = 1 \rightarrow NP$ **do**
      for $j = 1 \rightarrow D$ **do**
        $OP_{G,i,j} = 2 \times M_j - P_{G,i,j}$
      end for
    end for
  // End of centroid opposite population calculation
    Randomly initialize opposite points between M_i and the boundary point in the opposite direction if any opposite points are outside the search space.
    Select NP fittest individuals from the set $\{P_0 \cup OP_0\}$ as current population
  end if
  // End of opposition-based generation jumping
end while

B. Opposition-Based Generation Jumping

The similar approach of opposition-based population initialization is applied in this stage, based on a probability value called the generation jumping rate, Jr. So NP fittest individuals are selected from $\{P \cup \breve{P}\}$; P and $\breve{P}$ indicate the current and opposite population, respectively. The optimum value for Jr lies within the range $[0, 0.4]$ [10], and similar to DEs other control parameters, the optimal value of Jr is problem dependent [10]. When a centroid-opposite point falls outside the search space, the opposite point needs to be re-sampled to a random point so by this way it remains a valid opposite point. This is achieved by the following scheme. Having considered the search space as $[a, b]$ and the centroid point as $\mathbf{M}$, the opposition point for the whole space is calculated as

$$\breve{x} = d + rand(0, 1) \times (c - d), \tag{20}$$

where

$$d = \begin{cases} \mathbf{M}, & \breve{x} > b \\ a, & \breve{x} < a \end{cases}, \tag{21}$$

and

$$c = \begin{cases} b, & \breve{x} > b \\ \mathbf{M}, & \breve{x} < a \end{cases}. \tag{22}$$

The detailed description of the CODE algorithm is provided in Algorithm 1.

V. Experimental Results

In this section, the CODE is compared with its two parent algorithms (i.e., ODE and DE) and then with state-of-the-art DE-based algorithms, namely SaDE [9], ADE [27], SDE [12], and jDE [13]. The comparisons are based on solution accuracy. First, the employed benchmark functions will be explained and then control parameter settings for the algorithms.

A. Parameter Settings and Benchmark Functions

The comparisons are conducted on 24 well-known benchmark functions which are selected from the CEC-2005 special session on real parameter optimization [28] and the CEC-2008 special session and competition on large-scale global optimization [29]. To make the problems more complicated, they are rotated and shifted. The rotation is achieved by orthogonal matrices while the shifting is retained as before. Since the test suite involves two sessions (CEC-2005 and CEC-2008) the composition of the benchmark functions is summarized in Table I by indicating the corresponding explicit session and function type.

The mutation strategy and parameter settings are chosen based on the previous work as follow:

- Population size, $NP = 10 \times D$ [30]
- Differential amplification factor, $F = 0.5$ [2], [4], [6], [30]
- Crossover rate, $Cr = 0.9$ [2], [4], [6], [30]
- Generation jumping rate (for ODE and CODE), $Jr = 0.3$ [10]

TABLE I
Set of 24 benchmark functions from CEC 2005 and CEC 2008 competition sessions. All functions are shifted except F6 and F11. All functions are rotated orthogonally.

F	Benchmark Function	CEC Session	Function Type
F1	Sphere Function	F1-2008	Unimodal
F2	Schwefels Problem 2.21	F2-2008	Unimodal
F3	Schwefels Problem 1.2	F2-2005	Unimodal
F4	High Conditioned Elliptic Function	F3-2005	Unimodal
F5	Schwefels Problem 1.2 with Noise in Fitness	F4-2005	Unimodal
F6	Schwefels Problem 2.6 with Global Optimum on Bounds	F5-2005	Unimodal
F7	Rosenbrocks Function	F3-2008	Multimodal
F8	Rastrigins Function	F4-2008	Multimodal
F9	Riewanks Function	F5-2008	Multimodal
F10	Ackleys Function	F6-2008	Multimodal
F11	FastFractal DoubleDip Function	F7-2008	Multimodal
F12	Weierstrass Function	F11-2005	Multimodal
F13	Schwefels Problem 2.13	F12-2005	Multimodal
F14	Expanded Griewanks plus Rosenbrocks Function	F13-2005	Expanded
F15	Expanded Scaffers F6 Function	F14-2005	Expanded
F16	Hybrid Composition Function	F16-2005	Hybrid
F17	F16 with Noise in Fitness	F17-2005	Hybrid
F18	Hybrid Composition Function	F18-2005	Hybrid
F19	F18 with narrow basin global optimum	F19-2005	Hybrid
F20	F18 with global optimum on the bounds	F20-2005	Hybrid
F21	Hybrid Composition Function	F21-2005	Hybrid
F22	Non-Continuous version of F21	F23-2005	Hybrid
F23	Hybrid Composition Function	F24-2005	Hybrid
F24	F23 without bounds	F25-2005	Hybrid

- Mutation strategy DE/rand/1/bin (classical version) [2], [4], [30]
- Maximum number of function evaluations, $Max_{NFC} = 5000 \times D$
- Value to reach, $VTR = 1e - 8$

B. Simulation Results

1) Comparison of DE, ODE, and CODE: Each test function has been evaluated by 30 independent runs per algorithm to report mean, standard deviation, best, and the worst error values (i.e., $|f(\mathbf{x}) - f(\mathbf{x}^*)|$). The results of experiments for 100-dimensional problems are reported in Table II-Table IV. In these tables, bold faced results differentiate the algorithm that yields better solution accuracy than the other two counterparts. As shown in Table II and Table III for D=100, while the proposed CODE algorithm showed better solution accuracy in 19 instances out of 24, ODE and DE did better on 4 and 1 instances, respectively. Furthermore, ODE performed in a similar way in by surpassing DE and CODE for functions F2, F11, F15, and F24.

2) Comparison of CODE with state-of-the-art DE algorithms (i.e., SaDE, ADE, SDE, and jDE): The main contribution in this paper is to enhance ODE by introducing a new scheme of opposition and not outperforming the state-of-the-art algorithms, especially their adaptive variants. Furthermore, the investigation of a centroid-opposition based version of them would be worthwhile. In addition, compar-

TABLE II
RESULTS FOR 100D PROBLEMS (F1-F12)

F	Measure	DE	ODE	CODE
1	Mean	2.64E+04	3.71E+04	**3.88E-01**
	Std	2.35E+03	3.22E+03	**8.23E-01**
	Best	2.22E+04	2.70E+04	**3.90E-03**
	Median	2.59E+04	3.72E+04	**1.06E-01**
	Worst	3.14E+04	4.17E+04	**4.2412**
2	Mean	7.11E+01	**2.77E-01**	3.53E+01
	Std	4.4473	**1.4011**	9.1469
	Best	6.37E+01	**4.00E-03**	1.66E+01
	Median	7.07E+01	**2.00E-02**	3.41E+01
	Worst	8.11E+01	**7.6952**	5.14E+01
3	Mean	4.18E+05	4.39E+05	**4.49E+04**
	Std	2.34E+04	2.67E+04	**6.39E+03**
	Best	3.65E+05	3.81E+05	**3.61E+04**
	Median	4.18E+05	4.39E+05	**4.51E+04**
	Worst	4.63E+05	4.99E+05	**5.61E+04**
4	Mean	9.99E+07	1.57E+08	**5.14E+05**
	Std	1.03E+07	2.57E+07	**2.40E+05**
	Best	7.36E+07	8.78E+07	**1.76E+05**
	Median	1.02E+08	1.59E+08	**4.84E+05**
	Worst	1.19E+08	1.93E+08	**1.27E+06**
5	Mean	4.53E+05	4.91E+05	**5.67E+04**
	Std	4.57E+04	3.09E+04	**7.07E+03**
	Best	3.23E+05	4.23E+05	**4.40E+04**
	Median	4.56E+05	4.90E+05	**5.66E+04**
	Worst	5.24E+05	5.48E+05	**7.01E+04**
6	Mean	4.44E+04	4.64E+04	**1.29E+04**
	Std	1.55E+03	2.09E+03	**2.12E+03**
	Best	4.13E+04	4.17E+04	**9.72E+03**
	Median	4.45E+04	4.69E+04	**1.24E+04**
	Worst	4.72E+04	4.92E+04	**1.87E+04**
7	Mean	3.60E+09	2.26E+09	**4.08E+03**
	Std	6.54E+08	2.42E+09	**5.14E+03**
	Best	2.08E+09	9.64E+02	**4.92E+02**
	Median	3.72E+09	1.48E+09	**1.60E+03**
	Worst	4.66E+09	6.83E+09	**2.44E+04**
8	Mean	1.02E+03	1.05E+03	**3.99E+02**
	Std	2.71E+01	3.19E+01	**2.49E+02**
	Best	9.37E+02	9.74E+02	**5.76E+01**
	Median	1.02E+03	1.05E+03	**5.43E+02**
	Worst	1.06E+03	1.10E+03	**6.74E+02**
9	Mean	2.40E+02	3.44E+02	**1.23E-01**
	Std	2.62E+01	3.24E+01	**1.38E-01**
	Best	1.70E+02	2.77E+02	**8.30E-03**
	Median	2.42E+02	3.43E+02	**8.39E-02**
	Worst	2.91E+02	4.05E+02	**6.36E-01**
10	Mean	1.49E+01	1.50E+01	**6.14E-01**
	Std	4.11E-01	2.3743	**5.71E-01**
	Best	1.37E+01	2.651	**9.80E-03**
	Median	1.50E+01	1.54E+01	**3.67E-01**
	Worst	1.55E+01	1.67E+01	**1.9092**
11	Mean	-8.67E+02	**-9.69E+02**	-8.95E+02
	Std	1.23E+01	**1.59E+01**	1.70E+01
	Best	-8.91E+02	**-1.01E+03**	-9.48E+02
	Median	-8.67E+02	**-9.64E+02**	-8.93E+02
	Worst	-8.46E+02	**-9.44E+02**	-8.69E+02
12	Mean	1.38E+02	1.26E+02	**9.8313**
	Std	3.0908	5.0757	**4.1136**
	Best	1.32E+02	1.16E+02	**2.6565**
	Median	1.38E+02	1.27E+02	**8.3303**
	Worst	1.43E+02	1.40E+02	**2.04E+01**

TABLE III
RESULTS FOR 100D PROBLEMS (F13-F24)

F	Measure	DE	ODE	CODE
13	Mean	1.95E+07	1.10E+07	**5.96E+05**
	Std	7.97E+05	1.56E+06	**1.65E+05**
	Best	1.77E+07	9.21E+06	**2.52E+05**
	Median	1.96E+07	1.07E+07	**5.81E+05**
	Worst	2.09E+07	1.50E+07	**8.82E+05**
14	Mean	1.16E+05	7.12E+01	**4.47E+01**
	Std	4.08E+04	2.1939	**1.34E+01**
	Best	4.67E+04	6.57E+01	**1.34E+01**
	Median	1.04E+05	7.14E+01	**4.87E+01**
	Worst	2.22E+05	7.47E+01	**6.03E+01**
15	Mean	4.66E+01	**4.48E+01**	4.65E+01
	Std	2.20E-01	**4.42E-01**	2.53E-01
	Best	4.62E+01	**4.39E+01**	4.57E+01
	Median	4.67E+01	**4.48E+01**	4.65E+01
	Worst	4.69E+01	**4.56E+01**	4.70E+01
16	Mean	8.89E+02	9.19E+02	**3.34E+02**
	Std	1.60E+01	1.75E+01	**1.17E+01**
	Best	8.53E+02	8.89E+02	**3.18E+02**
	Median	8.90E+02	9.21E+02	**3.31E+02**
	Worst	9.20E+02	9.54E+02	**3.58E+02**
17	Mean	1.04E+03	1.05E+03	**4.86E+02**
	Std	1.17E+01	2.42E+01	**1.56E+01**
	Best	1.01E+03	9.85E+02	**4.55E+02**
	Median	1.04E+03	1.06E+03	**4.85E+02**
	Worst	1.06E+03	1.08E+03	**5.13E+02**
18	Mean	1.09E+03	9.01E+02	**9.01E+02**
	Std	9.3154	3.98E-01	**1.5222**
	Best	1.07E+03	9.00E+02	**9.00E+02**
	Median	1.09E+03	9.01E+02	**9.00E+02**
	Worst	1.12E+03	9.02E+02	**9.07E+02**
19	Mean	1.09E+03	9.01E+02	**9.00E+02**
	Std	1.51E+01	7.18E-01	**2.91E-01**
	Best	1.05E+03	9.00E+02	**9.00E+02**
	Median	1.09E+03	9.01E+02	**9.00E+02**
	Worst	1.12E+03	9.04E+02	**9.02E+02**
20	Mean	1.09E+03	9.01E+02	**9.00E+02**
	Std	1.45E+01	1.6344	**5.82E-01**
	Best	1.05E+03	9.00E+02	**9.00E+02**
	Median	1.09E+03	9.01E+02	**9.00E+02**
	Worst	1.12E+03	9.07E+02	**9.03E+02**
21	Mean	8.94E+02	8.97E+02	**5.00E+02**
	Std	9.5376	9.2578	**6.98E-02**
	Best	8.80E+02	8.76E+02	**5.00E+02**
	Median	8.94E+02	8.97E+02	**5.00E+02**
	Worst	9.19E+02	9.11E+02	**5.00E+02**
22	Mean	8.98E+02	8.95E+02	**5.19E+02**
	Std	8.5401	8.9486	**1.6878**
	Best	8.74E+02	8.74E+02	**5.18E+02**
	Median	8.99E+02	8.97E+02	**5.18E+02**
	Worst	9.10E+02	9.13E+02	**5.25E+02**
23	Mean	**1.28E+05**	1.28E+05	1.28E+05
	Std	**6.8244**	1.62E+01	7.00E+01
	Best	**1.28E+05**	1.28E+05	1.28E+05
	Median	**1.28E+05**	1.28E+05	1.28E+05
	Worst	**1.28E+05**	1.28E+05	1.28E+05
24	Mean	8.78E+02	**8.24E+02**	8.50E+02
	Std	1.36E+01	**1.20E+01**	1.19E+01
	Best	8.36E+02	**7.97E+02**	8.16E+02
	Median	8.81E+02	**8.25E+02**	8.49E+02
	Worst	9.01E+02	**8.52E+02**	8.71E+02

ative experiments will be conducted in this section. It is not feat to compare CODE with adaptive/self adaptive variants of DE. Purpose is just to assess the pros and cons of the proposed CODE method. The high computational cost of manual tuning of control parameters (CPs) of DE is very well understood in the DE community. Several DE-based variants were proposed to choose control parameters and trial vector generation strategies self-adaptively to achieve better performance compared to any of the manually-tuned DEs [9], [27], [31], [32]. In Ref. [9], the DE-based state-of-the-art algorithms such as SaDE [9], ADE [31], SDE [32], and jDE [27] were tested on the over set of a benchmark function. It was shown that SaDE performs better than any of its counterparts. The results obtained for SaDE, ADE, SDE, jDE in Ref. [9] are directly used in this section of the paper in determining the capability of CODE.

The set of benchmark functions tested for this purpose can be found In Ref. [9], in which functions F1 -F14 have been selected with the following setups.

- Problem dimensions: 10
- Mutation strategy for CODE algorithms: DE/rand/1/bin
- Differential amplification factor, $F = 0.9$
- Crossover rate, $Cr = 0.9$
- Value-to-reach, $VTR = 1e - 5$ [9]
- Maximum Number of Function evaluations: $1e + 5 \times D$ [9]
- Number of trial runs: 30 independent runs [9]

The numerical results are given in TABLE IV. In order to make the comparison easier, the results are summarized in TABLE V, where based on the total winning items they have been ranked. The results show that the SaDE and CODE are ranked first and second (shared with jDE), respectively.

TABLE IV
NUMERICAL RESULTS OF CODE, SADE, ADE, SDE, AND JDE FOR F1-F14 FUNCTIONS. SR INDICATES THE SUCCESS RATE. M INDICATES THE MEASURE. BEST RESULT FOR EACH SPECIFIED TEST FUNCTION (POSSIBLY SHARES WITH OTHER ALGORITHMS ALSO) IS IN BOLDFACE.

F	M	CODE	SaDE	ADE	SDE	jDE
1	Mean	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**
2	Mean	**0**	**0**	1.4E-4	**0**	0
	Std	**0**	**0**	2.4E-4	**0**	0
	SR	**100%**	**100%**	100%	**100%**	100%
3	Mean	**0**	**0**	1.5E+00	2.0E+00	1.3E-13
	Std	**0**	**0**	2.6E+00	1.6E+00	7.3-13
	SR	**100%**	**100%**	0%	0%	100%
4	Mean	**0**	**0**	7.0E-02	**0**	0
	Std	**0**	**0**	5.8E-2	**0**	0
	SR	**100%**	**100%**	0%	**100%**	100%
5	Mean	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**
6	Mean	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**
	SR	**100%**	**100%**	**100%**	**100%**	**100%**
7	Mean	2.1E-02	**0**	2.5E-07	7.3E-03	5.7E-04
	Std	2.7E-02	**0**	1.4E-06	7.5E-03	2.2E-03
	SR	30%	**100%**	100%	40%	93%
8	Mean	1.9E-02	**1.3E-02**	7.9E-02	3.8E-02	2.2E-02
	Std	1.4E-02	**1.1E-02**	4.2E-02	3.0E-02	1.7E-02
	SR	100%	**20%**	0%	0%	07%
9	Mean	4.2E+00	**0**	**0**	6.9E-01	**0**
	Std	3.8E+00	**0**	**0**	8.7E-01	**0**
	SR	20%	**100%**	**100%**	50%	**100%**
10	Mean	4.5E+00	**3.8E+00**	9.4E+00	7.7E+00	5.7E+00
	Std	2.6E+00	**1.3E+00**	2.2E+00	3.1E+00	2.1E+00
	SR	0%	**0%**	0%	0%	0%
11	Mean	5.62E+00	**0**	**0**	1.22E+00	**0**
	Std	2.3E+00	**0**	**0**	9.96E-01	**0**
	SR	0%	**100%**	**100%**	27%	**100%**
12	Mean	9.5E+01	**0**	**0**	2.3E+01	**0**
	Std	1.9E+02	**0**	**0**	4.8E+01	**0**
	SR	70%	**100%**	**100%**	80%	**100%**
13	Mean	**0**	**0**	1.4E-01	3.0E+01	1.3E+01
	Std	**0**	**0**	5.7E-01	4.6E+00	3.4E+01
	SR	**100%**	**100%**	90%	70%	87%
14	Mean	**1.7E-01**	2.5E-01	5.8E+00	8.2E+00	1.2E+00
	Std	**4.6E-01**	5.2E-01	4.8E+00	2.5E+01	3.2E+00
	SR	**87%**	80%	0%	23%	53%

TABLE V
SUMMARIZED COMPARISON RESULTS FROM TABLE IV.

	CODE	SaDE	ADE	SDE	jDE
Total	8	13	6	5	8
Rank	2	1	3	4	2

VI. CONCLUSION AND FUTURE WORKS

Similar to multiple evolutionary algorithms, DE can suffer from slow and/or premature convergence depending on the complexity of the problem. Also, its sensitiveness to the control parameters have eluded researchers from all over the world. They have been solved to a certain extent by adopting different mutation strategies and self-adaptive tuning of control parameters. Although incorporating opposition-based learning in DE has proven to be efficient for most problems, there is still much room for improvement. This paper provided a new OBC scheme to enhance the conventional OBC. The conventional OBC scheme in ODE has slower learning from an opposite candidate solution to its candidate counterpart because it uses extreme members maximum and minimum of the population. But the new OBC scheme known as centroid OBC accounts achieves stronger learning properties by considering all members of the population in the learning process.

The new opposition scheme with a new variant of ODE called CODE is evaluated using well-known challenging benchmark functions. The performance of CODE was measured in terms of solution accuracy and compared with its parents, DE and ODE. Furthermore, CODE's competiveness to state-of-the-art DE algorithms was evaluated. The results showed that CODE significantly performs better than DE and ODE for 100D problems and it showed promising results (ranked second) when compared to SaDE, ADE, SDE, and jDE algorithms. The main contribution of this paper is to enhance opposition-based computation schemes by giving an equal role to all individuals in the current population. The previous version worked with only two extreme points (i.e., Min and Max).

Future research will propose variant centroid-opposition based mutation strategies and also a self-adaptive CODE. This can target the effects of the new scheme over all opposition-based algorithms which are proposed in the soft computing field.

REFERENCES

[1] R. Storn and K. Price, "Differential evolutionA simple and efficient adaptive scheme for global optimization over continuous spaces," Berkeley, CA, Tech. Rep TR-95-012, 1995.

[2] R. Storn and K. Price, "Differential evolution-A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, 1997, pp. 341-359.

[3] S. Das and P.N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions On Evolutionary Computation*, vol. 15, no. 1, 2011, pp. 4-31.

[4] K. Price, R. Storn, and J. Lampinen, "Differential Evolution-A Practical Approach to Global Optimization," Berlin, Germany: Springer, 2005.

[5] K. V. Price, "An introduction to differential evolution," New Ideas in Optimization, D. Corne, M. Dorigo, and V. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 79-108.

[6] M.M. Ali and A. Trn, "Population set based global optimization algorithms: Some modifications and numerical studies," *Computer Operational Research*, vol. 31, no. 10, pp. 1703-1725, 2004.

[7] J. Ronkkonen, S. Kukkonen, and K.V. Price, "Real parameter optimization with differential evolution," in*Proc. IEEE World Congress on Computational Intelligence*, vol. 1, 2005, pp. 506-513.

[8] A.E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transaction in Evolutionary Computation*, vol. 3, no. 2, pp.124-141, 1999.

[9] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transaction in Evolutionary Computation*, vol. 13, no. 2, pp. 398-417, Apr. 2009.

[10] S. Rahnamayan, H.R. Tizhoosh, and M.M.A. Salama, "Opposition-based differential evolution," *IEEE Transaction in Evolutionary Computation*, vol. 12, no.1, pp.64-79, Feb. 2008.

[11] H.R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proc. Int. Conf. Comput. Intell. Modelling Control and Autom.*, Vienna, Austria, 2005, vol. I, pp. 695-701.

[12] H.R. Tizhoosh, "Opposition-Based Learning: A New Scheme for Machine Intelligence," in *Proc. Int. Conf. on Computational Intelligence for Modelling Control and Automation*, Vienna, Austria, Vol. I, 2005, pp. 695-701.

[13] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, "Opposition-Based Differential Evolution Algorithms," in *Proc. IEEE World Congress on Computational Intelligence*, Vancouver, Canada, 2006, pp. 7363-7370.

[14] A. R. Malisia and H. R. Tizhoosh, "Applying opposition-based ideas to the ant colony system," in *IEEE Swarm Intelligence Symposium*, 2007, pp. 182-189.

[15] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, "Opposition-Based Differential Evolution for Optimization of Noisy Problems," in *Proc. IEEE World Congress on Computational Intelligence*, Vancouver, Canada, 2006, pp. 6756-6763.

[16] S. Rahnamayan, G. Gary Wang, "Investigating in Scalability of Opposition- Based Differential Evolution," in *Proc. 8th WSEAS International Conference on Simulation, Modeling and Optimization*, Santander, Cantabria, Spain, September 23-25, 2008, pp. 105-111.

[17] R. Balamurugan, S. Subramanian, "Emission-constrained Dynamic Economic Dispatch using Opposition-based Self-adaptive Differential Evolution Algorithm,"*International Energy Journal*, Vol. 10, Issue 4, December 2009.

[18] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, "Quasi-Oppositional Differential Evolution," in *IEEE Congress on Evolutionary Computation*, Singapore, Sep. 2007, pp. 2229-2236.

[19] L. Peng, Y. Wang, "Differential Evolution using Uniform-Quasi-Opposition for initializing the Population," *Information Technology Journal*, vol.9, no.8, 2010, pp. 1629-1634.

[20] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, "Opposition-Based Differential Evolution (ODE) With Variable Jumping Rate," in *Proc. of IEEE Symposium on Foundations of Computational Intelligence*, Honolulu, Hawaii, USA, April 2007, pp. 81-88.

[21] M. Weber, V. Tirronen, F. Neri, "Scale Factor Inheritance Mechanism in Distributed Differential Evolution," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, Springer, vol. 14, Issue 11, Sept. 2010, pp. 1187-1207.

[22] F. Neri, V. Tirronen, "Recent Advances in Differential Evolution: A Review and Experimental Analysis," *Artificial Intelligence Review*, Springer, vol. 33, Issue 1, Feb. 2010, pp. 61-106.

[23] J. Brest, A. Zamuda, B. Bokovi, M. Maucec, V. umer, "High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction," in *Proc. IEEE World Congress on Computational Intelligence*, 2008, pp. 2032-2039.

[24] S. Das, A. Abraham, UK Chakraborty, "Differential Evolution with a Neighborhood-based Mutation Operator," in *Proc. of the IEEE Congress on Evolutionary Computation*, 2009, pp. 526-553.

[25] H. Salehinejad, S. Talebi, and F. Pouladi, "A metaheuristic approach to spectrum assignment for opportunistic spectrum access," in *Proc. IEEE 17th International Conference on Telecommunications*, 2010, pp. 234-238.

[26] J. Lampinen, I. Zelinka, "On Stagnation of the Differential Evolution Algorithm," in *Proc. 6th international Mendel conference on soft computing*, 2000, pp. 76-83.

[27] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transaction in Evolutionary Computation*, vol. 10, no. 6, pp. 646-657, Dec. 2006.

[28] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S.Tiwari, "Problem definitions and criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technol. Univ., Singapore, Tech. Rep., No. 2005005, May 2005, IIT Kanpur, India.

[29] K. Tang, X. Yao, P.N. Suganthan, C. MacNish, Y.P. Chen, C.M. Chen and Z. Yang, "Benchmark functions for the CEC 2008 special session and competition on large scale global optimization," Nature Inspired Comput. Applicat. Lab., USTC, China, Nanyang Technol. Univ., Sinagapore, Tech. Rep., 2007.

[30] H. Salehinejad and S. Talebi, "Dynamic Fuzzy Logic-Ant Colony System-Based Route Selection System," *Applied Computational Intelligence and Soft Computing*, vol. 2010, Article ID 428270, 13 pages, 2010.

[31] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *Proc. Mendel 9th Int. Conf. Soft Comput.*, Jun. 2003, pp. 41-46.

[32] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive differential evolution," Lecture Notes in Artificial Intelligence. Berlin, Germany: Springer-Verlag, 2005, pp. 192-199.

[33] A. R. Iqbal, M.A., Khan, N.K., Mujtaba, and H., Baig, "A novel function optimization approach using opposition based genetic algorithm with gene excitation," *Int. J. Innov. Comput. Inf. Control*, vol. 7, no. 7, pp. 4263-4276, 2011.