Enhanced Differential Evolution Using Center-Based Sampling

Ali Esmailzadeh, *IEEE Member*
Faculty of Engineering and Applied Science
University of Ontario Institute of Technology (UOIT)
Oshawa, Ontario L1H 7K4
Email: ali.esmailzadeh@uoit.ca

Shahryar Rahnamayan, *IEEE Member*
Faculty of Engineering and Applied Science
University of Ontario Institute of Technology (UOIT)
Oshawa, Ontario L1H 7K4
Email: shahryar.rahnamayan@uoit.ca

***Abstract*—The classical Differential Evolution (DE) has showed to perform efficiently in solving both benchmark functions and real-world problems. However, DE, similar to other evolutionary algorithms deteriorate in performance during solving high-dimensional problems. Opposition-based Differential Evolution (ODE) was introduced and, in general, has shown better performance comparing to classical DE for solving large-scale problems. In this paper, we propose an enhancement to ODE in order to improve its ability to solve large-scale problems more effectively. The proposed modified version of ODE is called Center-Based Differential Evolution (CDE) which utilizes the exact algorithm of ODE except replacing of opposite points with *center-based* individuals. This paper compares DE and ODE with the proposed algorithm, CDE. Furthermore, CDE with dynamic range (CDE_d) will be compared to CDE with fixed range (CDE_f). Experimental verifications are conducted on seven well-known shifted large-scale benchmark functions for dimensions of 100 and 500, including detailed parameter analysis for CDE. The shifted version of the functions ensures there is no bias towards the center of search space, in favor of CDE algorithm. The results clearly show that the CDE outperforms DE and ODE during solving large-scale problems, and also clarifies the superiority of CDE_d to CDE_f.**

I. Introduction

Differential Evolution (DE) algorithm over the years of research has showed to be an efficient algorithm comparing to other population-based algorithms. However, similar to other evolutionary algorithms, the efficiency of this algorithm is degraded during tackling with large-scale problems. For these kind of problems, the efficiency of the algorithm decreases and it is subject to the *curse of dimensionality*. On the other hand, majority of real-world problems are high-dimensional and time budget in solving them is limited. Many engineering and scientific fields, such as Finite Elements, Structural Designs, etc. are bound to deal with large-scale problems. Solving large-scale problems require great amount of time and resources due to the computational load and complexity; therefore, any attempt in constructing an algorithm that enhances solving large-scale problems decreases time and resources needed to solve the problem.

In the recent years, there have been extensive research conducted in the area of accelerating evolutionary algorithms, such as Opposition-based Differential Evolution (ODE) which is one of the main focuses of this paper, as a parent algorithm. ODE has shown good performance comparing to classical DE in solving large-scale [3] and noisy optimization problems [6]. In this paper, we want to improve ODE by replacing the *opposite* points with the newly introduced promising points, called Center-based points.

The rest of this paper is organized as follows: the concept of Center-based Sampling is reviewed and discussed, as well, it is intuitively explained in Section II. The proposed algorithm is presented in Section III. The experimental results and analysis are given in Section IV. Finally, the paper is concluded in Section V.

II. Center-Based Sampling: A Review

In this section, we explain the Center-point and Center-based sampling regions. These regions were introduced by Rahnamayan and Wang [5]. They investigated the closeness of points in a search space from an unknown solution (black-box problem). They measured the Euclidean distances of the points to the unknown solution for the different dimensions. They utilized Monte-Carlo simulation by dividing the [a,b] search space interval into partitions of 10^{-3} step-sizes, to represent a fixed point in a given calculation. For each fixed-point, x, in each dimension $(1, 2, 3, ..., D)$, they repeat the following steps 10^6 times (i.e. trials) in order to get measurable results:

1) Generate a uniform-random point, r, and a random *unknown-solution*, s, in the search space [a,b].
2) Measure the Euclidean distance of the fixed-point and the uniform-random point, from the unknown-solution.
3) Depending on which distance measure is smaller, the appropriate distance variables are updated for calculating the average distance and probability of closeness, at the end of the 10^6 trials.

By Monte-Carlo simulations, they have found that the probability of points being closer to an *unknown solution* (in comparison to uniformly generated points over the entire space) is much greater towards the center of the search space. Given the interval of search space as [a,b], shown in Fig. 1, the center of the interval is formulated by Eq. 1 and Eq. 2, for dimensions 1 and n, as follows:

For 1D:

$$c = \frac{(a+b)}{2} \tag{1}$$

978-1-4244-7835-4/11/$26.00 ©2011 IEEE

For n-D:

$$c_i = \frac{(a_i + b_i)}{2} \tag{2}$$

where $i = 1, ..., D$ and D is the dimension of the problem.

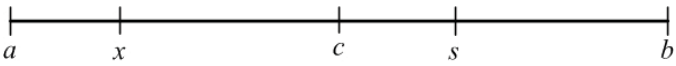

Fig. 1. The visual illustration (in 1D) of uniform-random point, x, and the unknown-solution, s, in the interval [a,b], where c indicates corresponding center of the search space, c=(a+b)/2.

They observed that as the candidate-solutions got closer to the center of the search space, the probability of closeness to the unknown solution rose drastically. Furthermore, they examined this phenomenon to solve high-dimensional problems [5]. Interestingly, as the dimensionality of the problem increased, the probability of closeness to the solution increases on the center of the search space, as well. They experimented this concept for large dimensions of 100, 200, 500 and 1000. Even for low dimension of one, the probability of closeness to the solution was greater around the center-point. In fact, according to Fig. 2 [5], there is a specific range in which the probability started to rapidly increase, for the higher dimensions. That range was found out to be [0.2,0.8] in a [0,1] search space interval; which is 60% of the search space, which falls on the center of the interval. As the dimensionality of the problem increases, the probability becomes closer to 100%, as indicated in Fig. 2.

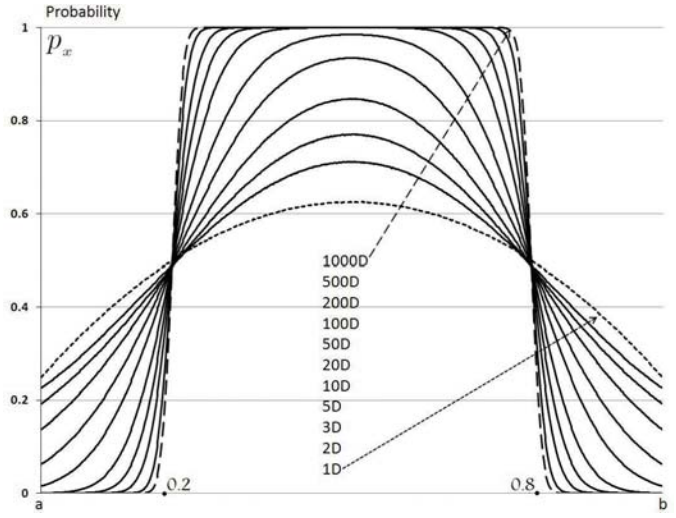

Fig. 2. Probability of closeness of candidate-solution to an unknown solution, for different interval points and dimensions [5].

Furthermore, the same improvements around the center-point can be seen with regards to the average distance of candidate-solutions from an unknown solution. In the Fig. 3, it can be seen that when the dimensionality of the problem increases, the average distance of candidate-solutions near and on the center-point decreases to very low values.

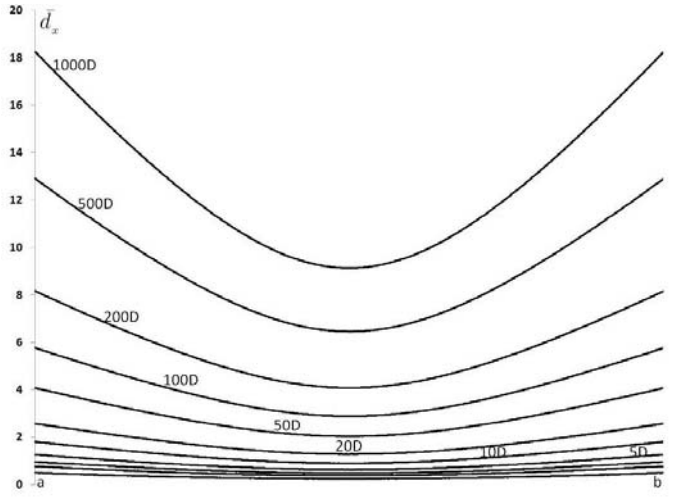

Fig. 3. The average distance of candidate-solution to an unknown solution, for different high dimensions [5].

However, as the authors have also mentioned, the center-point is a unique point; therefore, it can only be considered for s-metaheuristic algorithms, and is not usable for population-based algorithms [5].

As seen in Fig. 2, for all the various dimensions tested, the probability graphs start a sharp increase at two specific points in the range. The authors investigated this *magic* range further and found out that the specific points are 0.2 and 0.8 (in [0,1] interval), as indicated previously. They call the region in the interval [0.2,0.8] the *center-based* region. All the probability graphs, for all the different dimensions intersected at those two exact points in the range. In order to investigate and experiment further with the *center-based* region to find out the feature of points in that region, they ran the same set of experiments as with center-point, to find the probability of closeness of uniform-pseudorandom points generated in the *center-based* region, and compared them to random points generated in the entire range of [a,b] (in this case [0,1]). As seen in Fig. 4, the probability of closeness of candidate solutions generated in the *center-based* region, compared to random points in the entire range, is still very high; furthermore, as the dimension increases the probability gets closer to 1.

The phenomenon of center can be intuitively explained. According to Fig. 1, c is the middle of the search space, which has divided the entire search interval of [a,b] into sub-intervals of [a,c] and [c,b]. The candidate-solution x, and unknown solution s, can each be in different sub-intervals, or they can both be in the same sub-interval. The chances of either of the previous cases are 50% since we are dealing with uniform-random. For the former case, since c is in between x and s, no matter which sub-interval s belongs to, the Euclidean distance of x and c to s is $|x - s| \geq |c - s|$. Therefore, on average, in 50% of the times, c is always closer to s, than x is to s. Therefore, on 50% of the times, c is closer to the unknown solution. For the rest of the probability, where x and

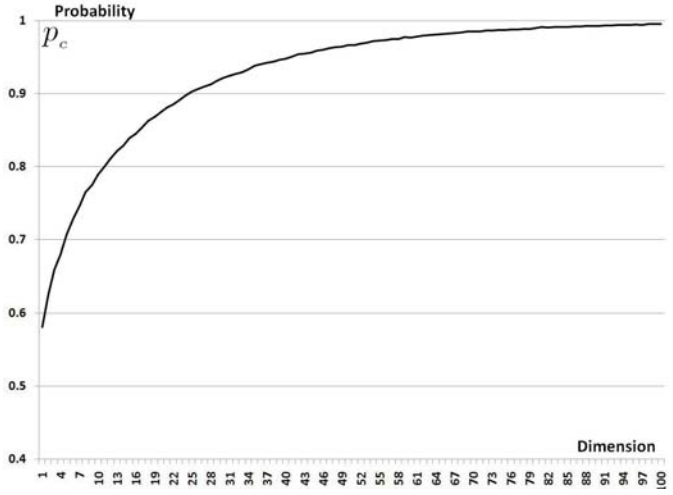

Fig. 4. Probability of closeness of candidate generated randomly in the center-based region, compared to random candidate generated in the entire range, for dimensions of 1 to 100. [5]

s are in the same sub-interval, then x and c are competing together for closeness to s. Therefore, any probability of c closer to s in this scenario, along with the 50% chance in the first scenario, will only help increasing the chances of c being closer to s, in overall. Furthermore, as the dimensionality increases, the rule of addition indicates that the probabilities of closeness of c to s will increase over the whole dimension, since the probabilities are added for each dimension of the problem. That explains why as the dimensionality increases, the probability of closeness to solution is the highest at the center of the space, as shown in Fig. 2.

It is important to distinguish between the *center-point* and *center-based* regions introduced and experimented in [5]. As previously mentioned, *center-point* cannot be used for p-metaheuristic algorithms since it considers only one single point and does not promote or support diversity in population-based algorithms. However, the *center-based* sampling can be utilized in p-metaheuristic algorithms since it can consider and include population generation of candidate-solutions in a specific range.

In the next section, we will examine the center-based concept further and will define a different (dynamic) version of the center-based region based on definition of the opposite point.

III. Center-Based Differential Evolution (CDE)

Before we introduce the proposed method, we the parent algorithm of the proposed algorithm. The Opposition-Based Learning (OBL) concept [1] was applied and tested on the classical DE algorithm by Rahnamayan, et al. [2], called Opposition-Based Differential Evolution (ODE). The DE algorithm is an evolutionary, population-based algorithm, which, similar to other evolutionary algorithms, an initial population of candidate-solutions is generated uniform randomly. The random population generation can play a role in term of convergence speed of an algorithm since the distance of the candidates from the unknown solution determines how fast an optimal solution can be found. In the ODE algorithm, the concept of *opposition* has been used to decrease the distance from unknown solution by comparing the candidate solution with its opposite and continuing with the better one. In steps of DE where a uniform-random population is generated, the opposites of each candidate-solution is calculated and populated. Then, from union of the current population and the opposite one, the fittest candidate-solutions are selected. Therefore, the current population includes uniform-random and opposite individuals with the better fitness values. Another additional modification to DE, by ODE is the generation jumping based on a new checking variable called generation jumping rate, J_r, which is constant value in the entire run, and determines opposition-based generation jumping rate. The flowchart of ODE is presented in Fig. 5.

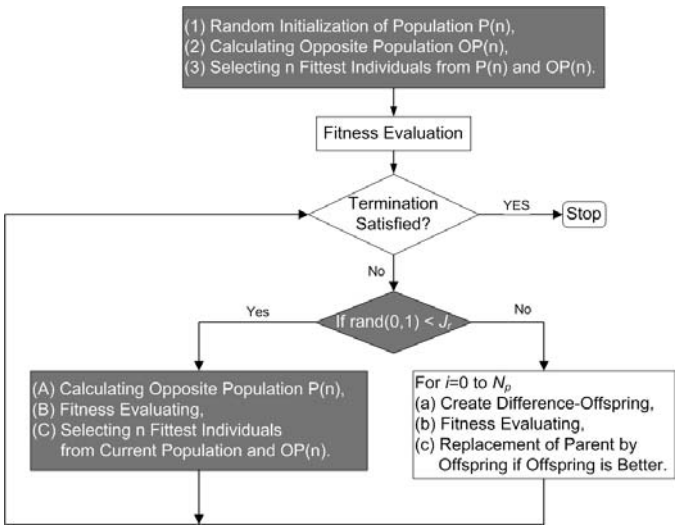

Fig. 5. The flowchart presentation of ODE; the gray boxes represent the components which are added or modified in the original DE algorithm. [6]

As discussed in Section II, the center-point region cannot be utilized for p-metaheuristic algorithms, since it is a unique point; therefore, it does not promote diversity required for population-based algorithms. On the other hand, the center-based sampling method deals with a specific range within the entire space, in which uniform pseudorandom points are generated; therefore, this sampling method can be utilized for population-based algorithms.

The center-based sampling region which was introduced in [5] and reviewed in Section II, considers a certain boundary for the interval of *center-based* region. More specifically, the sampling covers the points in [0.2,0.8] for interval of [0,1]. Since the range is set to exact boundaries, it is possible that this definition of center-based region would not be flexible enough, as a result, it might not be robust enough for variant problems. In this paper, we propose a slightly different definition for the center-based region. In our proposed algorithm, we keep the same scheme of ODE but we replace opposite points with the

uniform points which are generated randomly in the modified center-based interval. That region will be changed dynamically according to the corresponding opposite points.

In reviewing ODE, it was mentioned that for each uniform-randomly generated candidate-solution, an opposite candidate-solution was created, during both the population initialization and generation jumping. More specifically, the opposite point of candidate-solution x, is denoted $\hat{x}$ and it is the mirror reflection of x from the center of space, c. Since ODE was shown by previous research works to outperform DE in solving large-scale problems, we would like to use the same algorithm as a parent algorithm to propose CDE.

In our proposed method, we consider generating a uniform-random individual in the range between the current candidate solution and its *opposite* point. In other words, given the current candidate solution point of x, and the opposite of the current candidate point $\hat{x}$, then the new Center-based candidate-solution is a uniform-random point generated between x and $\hat{x}$. Mathematically, center-based population can be denoted as follows in Eq. 3:

For n-D:

From opposition concept and Eq.s 1, and 2 the following can be derived:

$$x_i^{cb} = \begin{cases} x_i + (\hat{x}_i - x_i) \times rand(0,1) & \text{if } x_i \leq c, \\ \hat{x}_i + (x_i - \hat{x}_i) \times rand(0,1) & \text{otherwise.} \end{cases} \quad (3)$$

where $i = 1, ..., D$ and D is the dimension of the problem.

We call this version of CDE as *CDE with Dynamic range* (CDE_d).

In order to illustrate this concept, we use the visual presentation of Center-based interval for 1D in Fig. 6. Moreover, the 2D presentation of center-based region is shown in Fig. 7. The same definition is used for higher dimensions of the search space.

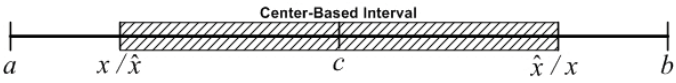

Fig. 6. The visualization of the proposed interval in 1D. The Center-based interval is between the candidate solution x and the opposite candidate solution, $\hat{x}$.

The utilized opposite points in ODE are replaced with the newly generated center-based point. If the convergence speed of CDE is improved, it is evident that this new segmentation of the search space and concept of center-based sampling is a better sampling interval choice.

The following steps explain population generation during population initialization and generation jumping in CDE_d:

1) Uniform-randomly generate a vector of population individuals in the range [a,b]; this is the *current* population (P(n)).
2) Calculate *opposite* of each of the corresponding individuals from current population; this forms the *opposite* population.

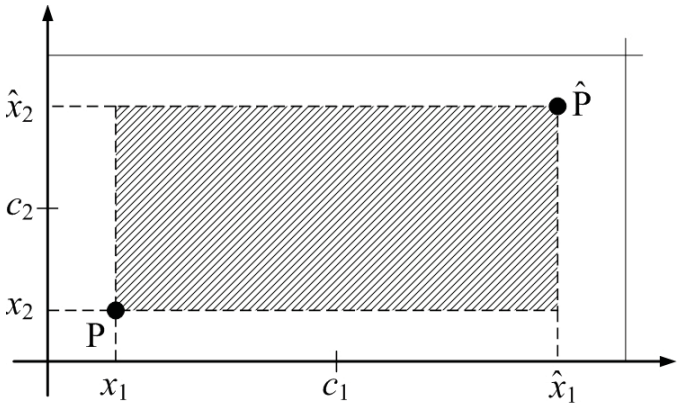

Fig. 7. The visualization of the proposed region in 2D. The Center-based region is shown by shadowed area.

3) For each dimension, in the interval between each individual from the *current* population and its corresponding *opposite* individual, generate a uniform-random point inside that range; save the generated individual in the *center-based* population (CB(n)).
4) Union the current and center-based population (P(n) and CB(n), respectively. Select n fittest individuals from the current set.
5) Continue with the rest of DE steps.

The above steps are also performed for CDE with the fixed range of 60% of the search space (from [5]), called *CDE with Fixed Range* (CDE_f). However, in CDE_f instead of generating points in the interval of [x,$\hat{x}$] as with CDE_d, we generate uniform-random points in the middle 60% interval of the search space. The CDE_f interval denoted as the range of [fa,fb], in search space of [a,b] can be calculated by Eq. 4, below.

$$\begin{aligned} fa_i &= a_i + 0.20 \times (b_i - a_i) \\ fb_i &= a_i + 0.80 \times (b_i - a_i) \end{aligned} \quad (4)$$

where $i = 1, ..., n$ and n is maximum dimension (D) of the problem.

The CDE_d algorithm is illustrated in Fig. 8. The opposite point is calculated based on maximum and minimum values of the variables in the current population.

Both the ODE and CDE algorithms have utilized *generation jumping*. The rate of jumping (J_r) is set manually based on trial experiments and recommended values. The rate is kept constant for each of the algorithms, throughout the whole experiments.

Ultimately, we want to compare the CDE_f algorithm with fixed interval of [fa,fb], which is the 60% middle of the search space (described in Eq. 4), to CDE_d which uses the dynamic interval of [x,$\hat{x}$]. We would like to see the performance of these two different schemes of CDE in solving problems.

In this paper, we aim to compare CDE_d to the classical DE and ODE in order to investigate the performance of Center-

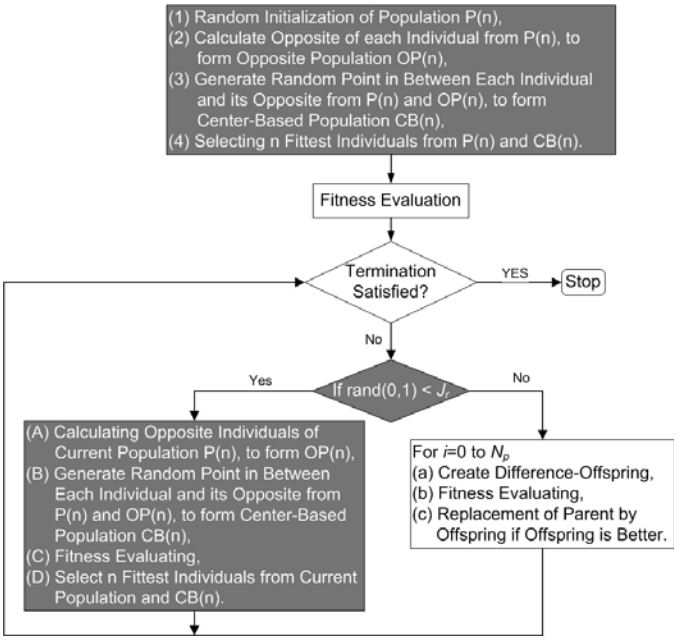

Fig. 8. Flowchart of the proposed CDE_d algorithm, made similar to the flowchart of ODE. The gray boxes represent addition or modifications to the parent DE algorithm.

based sampling on ODE. Furthermore, we compare CDE_d with CDE_f in order to analyze the effect of using dynamic range instead of a fix one.

IV. Experimental Verifications

In this section, we will investigate performance of the proposed algorithms CDE_d and CDE_f, against DE and ODE in terms of solution accuracy for $D = 100$ and $D = 500$. As well, we will perform parameter analysis on the J_r parameter in $D = 500$ for ODE and CDE_d, to obtain the optimal value for the J_r parameter.

A. Control Parameter Settings

All the common parameters of the DE, ODE, and CDE (both fixed and dynamic), are set to the same values as below, to have a fair comparison.

- Population size, $N_p = 100$
- Differential amplification factor, F=0.5
- Crossover probability constant, $C_r = 0.9$
- Strategy, $DE/rand/1/bin$
- Maximum number of function calls (termination criteria), MAX_{NFC}=$5000\times D$

B. Benchmark Functions

The utilized seven **shifted** benchmark functions are box-constrained high-dimensional problems for minimization as provided by the *CEC'2008 Special Session on Large Scale Global Optimization* [4]. The benchmark functions used in this paper are the ones used in [3] for large-scale problems. The functions $f1$ and $f2$ are unimodal and the rest of the functions are multimodal problems. The functions are randomly shifted to not have any kind of favor or bias towards the center of the search space, and towards the proposed method. Therefore, for none of the functions, the solution is located at the center.

C. Simulation Strategy

Similar to other studies in the evolutionary optimization [6], [8]–[10], for all conducted experiments, trials are repeated 25 times per function per dimension. Each run is continued up to $5000\times D$ function calls and then mean and standard deviation of the best fitness values are reported.

It is of great importance to note that even though for the definition of CDE_f algorithm, the interval was originally set as [0.2, 0.8] (for interval [0,1]) [5]; however, in conducting experiments, the *min* and *max* boundaries of each functions were taken into consideration.

The algorithms DE, ODE are compared once to CDE_f and once to CDE_d in dimensions of 100 and 500. As well, CDE_f and CDE_d algorithms are compared against each other in $D = 100$ and $D = 500$. In these experiments, the J_r values for ODE, CDE_f and CDE_d algorithms are set based on experiments done previously by other authors. The J_r value of ODE is set to 0.3, where as, both CDE_f and CDE_d algorithms have J_r value of 0.05. These J_r values are set based on [7].

Furthermore, since there is no detailed experiment done on J_r parameter analysis in ODE and CDE_d algorithms for $D > 100$, for a set of problems, we conduct a parameter analysis experiment to test each algorithm with different J_r values. From those results the most successful J_r value for each algorithm will be picked as the *benchmark* J_r values, which will be used to compare DE, ODE and CDE_d.

D. Results and Analysis

In this sub-section, there are two experimental series presented, and the results are analyzed. The first experimental series is concerned with experimenting the CDE_d and CDE_f methods to parent algorithms ODE and DE, for high dimensions, and conventional J_r parameter value. In the second experimental series, a detailed parameter analysis of J_r is done on ODE and CDE_d, for high dimension of 500. Furthermore, the *successful* J_r values for each algorithm are compared to each other.

Experimental Series 1: CDE_d and CDE_f

In this experimental series, DE, ODE, CDE_d, and CDE_f are compared for $D = 100$ and $D = 500$, with J_r values set conventionally by previous research work [7]. In these experiments, J_r value of ODE is 0.3 and for CDE (both versions), the J_r value is set to 0.05 [7]. The Tables I, II and III summarize the experimental results. Table I represents comparison among DE, ODE and CDE_d; Table II represents comparison of DE, ODE and CDE_f. Lastly, Table III represents comparison between the proposed algorithms CDE_d and CDE_f. All the tables show the result comparisons for both dimensions of 100 and 500.

For D=100, as illustrated in Table I, ODE is only successful on two out of the seven functions, $f2$ and $f4$. All 3 algorithms performs the same on $f7$ function. Furthermore, DE and CDE_d both outperform ODE on $f1$ and $f6$, but CDE_d outperforms the other two algorithms on one function, $f3$. These results indicate that in *lower* dimensions such as D=100, the proposed algorithm of CDE_d is only successful on three out of seven functions.

TABLE I
MEAN ± (STANDARD DEVIATION) OF THE BEST FITNESS VALUE OF DE COMPARED TO ODE AND CDE_d (DYNAMIC RANGE), FOR D=100 AND D=500. THE VALUES OF J_r PARAMETER FOR ODE AND CDE_d ARE 0.3 AND 0.05, RESPECTIVELY. THE BEST RESULT FOR EACH FUNCTION IS HIGHLIGHTED IN **BOLDFACE**.

	$D=100$		
F	DE	ODE	CDE_d
f_1	**0 ± (0)**	$0.02 \pm (0.13)$	**0 ± (0)**
f_2	$64.85 \pm (5.85)$	**29.19 ± (11.54)**	$65.03 \pm (6.57)$
f_3	$249.49 \pm (382.99)$	$1.46E6 \pm (5.26E6)$	**167.62 ± (83.77)**
f_4	$559.74 \pm (125.20)$	**396.97 ± (186.14)**	$569.40 \pm (113.05)$
f_5	**6.90E-4 ± (2.41E-3)**	$2.1E-2 \pm (0.049)$	$0.0015 \pm (0.0060)$
f_6	**0 ± (0)**	$0.72 \pm (0.52)$	**0 ± (0)**
f_7	$0 \pm (0)$	$0 \pm (0)$	$0 \pm (0)$
	$D=500$		
F	DE	ODE	CDE_d
f_1	**0 ± (0)**	$393.94 \pm (1271.35)$	**0 ± (0)**
f_2	$111.93 \pm (5.75)$	**80.65 ± (3.32)**	$95.75 \pm (1.22)$
f_3	$1828.49 \pm (458.30)$	$6.18E8 \pm (6.70E8)$	**1670.26 ± (226.17)**
f_4	$1868.76 \pm (135.15)$	$2556.85 \pm (145.26)$	**1630.90 ± (153.69)**
f_5	$0.16 \pm (0.27)$	$3.82 \pm (11.05)$	**0.05 ± (0.14)**
f_6	$7.62 \pm (1.44)$	$13.91 \pm (1.06)$	**7.25 ± (1.39)**
f_7	$0.06 \pm (0.27)$	$0.12 \pm (0.49)$	**0 ± (0)**

For high dimensional problems, such as D=500, according to results provided in Table I, CDE_d outperforms DE and ODE on five out of the seven benchmark functions. Only on $f2$ ODE performs better than CDE_d. Furthermore, DE and CDE_d both beat ODE on $f1$. Table I clearly indicates that the CDE_d algorithm outperforms the DE and ODE methods on the majority of benchmark functions, for large-scale problems of D=500.

Further to experimenting CDE_d, we compare CDE_f to its parent algorithms DE and ODE. In Table II, results of comparison are summarized. As the results indicate, for three of the functions, $f1$, $f3$ and $f6$, CDE_f outperforms ODE. Moreover, on two functions, $f1$ and $f6$, CDE_f and DE both had the same best result. Only on two functions $f2$ and $f4$, ODE performed better than the other algorithms. For high dimension of D=500, CDE_f outperforms DE and ODE on four out of seven functions. DE had the best performance of the other algorithms on two functions, $f1$ and $f3$, while ODE only outperformed others on one function, $f2$. These results clearly indicate that for large-scale problems, CDE_f algorithm performs better than DE and ODE on most of the functions.

The two proposed algorithms CDE_f and CDE_d, which are the winners of two previous tables, are compared against each other in Table III. For low dimension of D=100, CDE_f algorithm beats CDE_d on three of the seven functions, and both algorithms have the same best result on three of the

TABLE II
MEAN ± (STANDARD DEVIATION) OF THE BEST FITNESS VALUE OF DE COMPARED TO ODE AND CDE_f (FIXED RANGE), FOR D=100 AND D=500. THE VALUES OF J_r PARAMETER FOR ODE AND CDE_f ARE 0.3 AND 0.05, RESPECTIVELY. THE BEST RESULT FOR EACH FUNCTION IS HIGHLIGHTED IN **BOLDFACE**.

	$D=100$		
F	DE	ODE	CDE_f
f_1	**0 ± (0)**	$0.02 \pm (0.13)$	**0 ± (0)**
f_2	$64.85 \pm (5.85)$	**29.19 ± (11.54)**	$53.16 \pm (5.78)$
f_3	$249.49 \pm (382.99)$	$1.46E6 \pm (5.26E6)$	**133.49 ± (52.79)**
f_4	$559.74 \pm (125.20)$	**396.97 ± (186.14)**	$539.84 \pm (145.57)$
f_5	**6.90E-4 ± (2.41E-3)**	$2.1E-2 \pm (0.049)$	$0.0016 \pm (0.0046)$
f_6	**0 ± (0)**	$0.72 \pm (0.52)$	**0 ± (0)**
f_7	$0 \pm (0)$	$0 \pm (0)$	$0 \pm (0)$
	$D=500$		
F	DE	ODE	CDE_f
f_1	**0 ± (0)**	$393.9 \pm (1271.3)$	$3E-5 \pm (9E-5)$
f_2	$111.93 \pm (5.75)$	**80.65 ± (3.32)**	$98.11 \pm (3.34)$
f_3	**1828.49 ± (458.30)**	$6.18E8 \pm (6.70E8)$	$1833.53 \pm (348.32)$
f_4	$1868.76 \pm (135.15)$	$2556.85 \pm (145.26)$	**1470.08 ± (107.13)**
f_5	$0.16 \pm (0.27)$	$3.82 \pm (11.05)$	**0.14 ± (0.30)**
f_6	$7.62 \pm (1.44)$	$13.91 \pm (1.06)$	**6.72 ± (1.19)**
f_7	$0.06 \pm (0.27)$	$0.12 \pm (0.49)$	**7.77E-3 ± (0.033)**

functions, $f1$, $f6$ and $f7$. However, as indicated in Table III, for higher dimensions of D=500, CDE_d outperforms CDE_f on five out of the seven functions, $f1$, $f2$, $f3$, $f5$ and $f7$. These results clearly indicate that for large-scale problems, the CDE_d algorithm is the winner algorithm.

TABLE III
MEAN ± (STANDARD DEVIATION) OF THE BEST FITNESS VALUE OF CDE_f (FIXED RANGE) COMPARED TO CDE_d (DYNAMIC RANGE), FOR D=100 AND D=500. THE VALUES OF J_r PARAMETER FOR BOTH CDE_d AND CDE_f IS 0.05. THE BEST RESULT FOR EACH FUNCTION IS HIGHLIGHTED IN **BOLDFACE**.

	$D=100$	
F	CDE_f	CDE_d
f_1	$0 \pm (0)$	$0 \pm (0)$
f_2	**53.16 ± (5.78)**	$65.03 \pm (6.57)$
f_3	**133.49 ± (52.79)**	$167.62 \pm (83.77)$
f_4	**539.84 ± (145.57)**	$569.40 \pm (113.05)$
f_5	$1.57E-3 \pm (4.65E-3)$	**1.48E-3 ± (6.02E-3)**
f_6	$0 \pm (0)$	$0 \pm (0)$
f_7	$0 \pm (0)$	$0 \pm (0)$
	$D=500$	
F	CDE_f	CDE_d
f_1	$2.63E-5 \pm (8.69E-5)$	**0 ± (0)**
f_2	$98.11 \pm (3.34)$	**95.75 ± (1.22)**
f_3	$1833.53 \pm (348.32)$	**1670.26 ± (226.17)**
f_4	**1470.08 ± (107.13)**	$1630.90 \pm (153.69)$
f_5	$0.14 \pm (0.30)$	**0.05 ± (0.14)**
f_6	**6.72 ± (1.19)**	$7.25 \pm (1.39)$
f_7	$7.77E-3 \pm (0.033)$	**0 ± (0)**

To summarize the three tables, considering that the J_r parameter values are set based on previous research work by other authors, it is clear that for solving large-scale problems, the two proposed algorithms CDE_f and CDE_d outperform their parent algorithms, DE and ODE. Furthermore, CDE_d also outperforms the CDE_f algorithm over large scale problems.

Experimental Series 2: Parameter Analysis

This experimental series is concerned with parameter analysis of the J_r parameter values in the ODE and CDE_d algorithms. The different J_r parameter values are experimented for both ODE and CDE_d algorithms. The J_r parameter values for the ODE algorithm are summarized in Table IV. It is apparent that for the benchmark functions used in this section, it is better to use 0.05 setting for the J_r parameter of ODE, as the J_r=0.05 has dominated the other J_r values on five out of the seven functions. Furthermore, the same test was done for the CDE_d algorithm to determine the best J_r parameter value, which is summarized in Table V. The results are not as straight forward compared to the tests for ODE algorithm. However, it can be concluded that for CDE_d algorithm the better J_r parameter values to use is below 0.05 value.

Finally, based on the parameter analysis experiments done on the J_r parameter, for both ODE and CDE_d, the most *successful* J_r values for each of the algorithms are compared to each other to examine the competition of the successful J_r values against each other. This comparison is summarized in Table VI. The results indicate that overall, CDE_d based on its successful J_r values, has better performance than parent algorithms ODE and DE on four out of seven functions.

V. Concluding Remarks

The DE method is a well-known p-metaheuristic algorithm for solving challenging optimization problems. However, DE is subject to the *curse of dimensionality*, which its performance deteriorates as the dimensionality of the problem increases. The ODE algorithm has shown that it performs better than classical DE on solving large-scale problems. In a recent research work, the *center-point* and *center-based* sampling methods were introduced which have shown that the probability of closeness of a candidate-solution to an unknown solution, comparing to a random candidate in the entire search space, is at the highest at the center of search space. This is specially significant for high-dimensional problems. We have chosen DE algorithm in this paper since DE is a well-known global optimization algorithm, used in many research areas and is a fast and robust algorithm comparing to other population-based algorithms. In this paper, we proposed a new modification to the original ODE algorithm, and called it Center-Based Differential Evolution (CDE). In this method, we replaced opposite points of ODE with center-based points, which are generated in the *center-based* interval. Furthermore, we define two variants of CDE based on Dynamic range (CDE_d) and Fixed range (CDE_f) of center-based regions. The proposed algorithms were tested on seven well-known large-scale shifted benchmark functions to compare them with their parents, namely, DE and ODE. The purpose of testing on shifted functions was to ensure that there is no bias towards the center of the search space, and that the optimum solution can be randomly placed anywhere in the space. The CDE_f algorithm was compared to CDE_d, and the results confirmed that both CDE_f and CDE_d outperform DE and ODE on high dimensional problems of D=500. In addition, CDE_d outperforms CDE_f on solving large scale problems; whereas, CDE_f outperformed CDE_d for low dimensional problems, such as D=100.

Furthermore, we conducted a detailed parameter analysis of J_r parameter in both ODE and CDE_d algorithms, for dimension of $D = 500$. The results picked a winning J_r value for ODE and a set of winning J_r values for CDE_d. By comparing the winning J_r values for ODE and CDE_d, it has shown that CDE_d algorithm has outperformed both DE and ODE in four out of seven functions.

According to the simulations performed in this paper on the seven shifted benchmark functions, the results match with our expectation from earlier research, which indicate better results for CDE in solving large-scale problems because of using the center-based sampling.

As a future work, we will consider investigating the center-based concept (both dynamic and fixed) on other P-metaheuristic algorithms.

References

[1] H.R. Tizhoosh, *Opposition-Based Learning: A New Scheme for Machine Intelligence*, Int. Conf. on Computational Intelligence for Modelling Control and Automation (CIMCA'2005), Vienna, Austria, Vol. I, pp. 695-701, 2005.

[2] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Opposition-Based Differential Evolution*, IEEE Transactions on Evolutionary Computation, Volume 12, Issue 1, Feb. 2008, pp. 64-79.

[3] S. Rahnamayan, G. Gary Wang, *Solving Large Scale Optimization Problems by Opposition-Based Differential Evolution (ODE)*, WSEAS TRANSACTIONS on COMPUTERS Manuscript received May. 10, 2008; revised Sep. 18, 2008.

[4] K. Tang, X. Yao, P. N. Suganthan, C. Mac- Nish, Y. P. Chen, C. M. Chen, Z. Yang, *Benchmark Functions for the CEC2008 Special Session and Competition on Large Scale Global Optimization*, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php, 2007.

[5] S. Rahnamayan, G. Gary Wang, *Center-Based Sampling for Population-Based Algorithms*, IEEE Congress on Evolutionary Computation (CEC'09), pp. 933-938, May 2009.

[6] S. Rahnamayan, H. R. Tizhoosh, M. M.A. Salama, *Opposition-Based Differential Evolution for Optimization of Noisy Problems* IEEE Publications Proceedings of the Congress on Evolutionary Computation (CEC'06), pp. 6756-6763, 2006.

[7] S. Rahnamayan, H. R. Tizhoosh, M. M.A. Salama, *Quasi-Oppositional Differential Evolution*, IEEE Congress on Evolutionary Computation (CEC'07), Singapore, pp. 2229-2236, 2007.

[8] S. Das, A. Konar, Uday K. Chakraborty, *Improved Differential Evolution Algorithms for Handling Noisy Optimization Problems*, Proceedings of IEEE Congress on Evolutionary Computation, CEC2005, Vol. 2, pp. 1691-1698, 2005.

[9] T. Krink, B. Filipič, Gary B. Fogel, *Noisy optimization problems - A Particular Challenge for Differential Evolution?*, Proceedings of the 2004 Congress on Evolutionary Computation, CEC2004, Vol. 1, pp. 332-339, 2004.

[10] J. Vesterstrøm and R. Thomsen, *A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems*. Proceedings of the Congress on Evolutionary Computation (CEC'04), IEEE Publications, Vol. 2, pp. 1980-1987, 2004.

TABLE IV

MEAN ± (STANDARD DEVIATION) OF THE BEST FITNESS VALUE FOR J_r PARAMETER ANALYSIS OF ODE, FOR D=500. THE BEST RESULT FOR EACH FUNCTION IS HIGHLIGHTED IN **BOLDFACE**.

F	$J_r = 0.05$	$J_r = 0.10$	$J_r = 0.15$	$J_r = 0.20$
f_1	**3.52E-6 ± (4.273E-6)**	$7.757E-4 \pm (3.376E-3)$	$1.077E-3 \pm (2.105E-3)$	$1.249E-2 \pm (3.985E-2)$
f_2	$80.266 \pm (5.208)$	$79.943 \pm (5.551)$	$79.679 \pm (4.808)$	$79.546 \pm (4.053)$
f_3	**1575.803 ± (336.632)**	$1635.728 \pm (289.249)$	$1977.266 \pm (812.865)$	$1.018E8 \pm (2.241E8)$
f_4	**1608.951 ± (140.728)**	$1937.696 \pm (143.757)$	$2141.821 \pm (165.391)$	$2278.328 \pm (110.187)$
f_5	$0.489 \pm (1.066)$	$0.255 \pm (0.339)$	$0.435 \pm (0.975)$	**0.244 ± (0.325)**
f_6	**8.436 ± (1.528)**	$9.602 \pm (1.196)$	$11.137 \pm (0.824)$	$12.245 \pm (0.991)$
f_7	**1.784E-3 ± (2.609E-3)**	$2.172E-3 \pm (4.639E-3)$	$0.031 \pm (0.086)$	$0.012 \pm (0.036)$
F	$J_r = 0.25$	$J_r = 0.30$	$J_r = 0.35$	$J_r = 0.40$
f_1	$0.527 \pm (1.385)$	$393.937 \pm (1271.348)$	$934.727 \pm (1441.596)$	$2804.920 \pm (2946.949)$
f_2	**78.845 ± (3.025)**	$80.654 \pm (3.322)$	$81.449 \pm (3.687)$	$82.703 \pm (4.418)$
f_3	$4.500E8 \pm (8.650E8)$	$6.822E8 \pm (6.697E8)$	$9.085E8 \pm (9.051E8)$	$1.649E9 \pm (1.559E9)$
f_4	$2459.243 \pm (169.512)$	$2556.856 \pm (145.260)$	$2630.505 \pm (147.965)$	$2746.224 \pm (143.123)$
f_5	$0.474 \pm (0.871)$	$3.823 \pm (11.049)$	$10.623 \pm (16.083)$	$28.308 \pm (41.682)$
f_6	$13.226 \pm (0.872)$	$13.910 \pm (1.060)$	$15.512 \pm (1.098)$	$16.605 \pm (0.785)$
f_7	$0.070 \pm (0.197)$	$0.122 \pm (0.488)$	$0.045 \pm (0.096)$	$0.124 \pm (0.345)$

TABLE V

MEAN ± (STANDARD DEVIATION) OF THE BEST FITNESS VALUE FOR J_r PARAMETER ANALYSIS OF CDE_d, FOR D=500. THE BEST RESULT FOR EACH FUNCTION IS HIGHLIGHTED IN **BOLDFACE**.

F	$J_r = 0.01$	$J_r = 0.02$	$J_r = 0.03$	$J_r = 0.04$
f_1	$1.096E-5 \pm (3.43E-5)$	$1.80E-5 \pm (6.39E-5)$	$1.33E-4 \pm (6.53E-4)$	$8.72E-6 \pm (1.89E-5)$
f_2	$95.103 \pm (1.26)$	**95.089 ± (0.813)**	$95.278 \pm (0.921)$	$95.660 \pm (0.785)$
f_3	$1.789E3 \pm (4.429E2)$	$1.809E3 \pm (3.782E2)$	**1.627E3 ± (2.133E2)**	$1.699E3 \pm (3.074E2)$
f_4	**1.532E3 ± (1.382E2)**	$1.604E3 \pm (1.062E2)$	$1.635E3 \pm (1.457E2)$	$1.640E3 \pm (1.449E2)$
f_5	$0.111 \pm (0.212)$	$0.140 \pm (0.318)$	$0.188 \pm (0.307)$	$0.067 \pm (0.117)$
f_6	$6.660 \pm (1.137)$	**6.464 ± (1.010)**	$6.735 \pm (1.414)$	$6.709 \pm (1.042)$
f_7	$0 \pm (0)$	$0 \pm (0)$	$0 \pm (0)$	$0 \pm (0)$
F	$J_r = 0.05$	$J_r = 0.06$	$J_r = 0.07$	$J_r = 0.08$
f_1	**7.12E-6 ± (1.095E-5)**	$8.20E-6 \pm (1.55E-5)$	$1.13E-5 \pm (1.82E-5)$	$1.22E-5 \pm (1.91E-5)$
f_2	$95.749 \pm (1.219)$	$95.908 \pm (0.684)$	$95.836 \pm (0.663)$	$95.904 \pm (0.723)$
f_3	$1.670E3 \pm (2.262E2)$	$1.839E3 \pm (3.843E2)$	$1.774E3 \pm (3.419E2)$	$1.726E3 \pm (2.232E2)$
f_4	$1.630E3 \pm (1.537E2)$	$1.709E3 \pm (1.711E2)$	$1.694E3 \pm (2.539E2)$	$1.717E3 \pm (1.309E2)$
f_5	$0.051 \pm (0.142)$	$0.137 \pm (0.217)$	$0.160 \pm (0.310)$	$0.114 \pm (0.195)$
f_6	$7.255 \pm (1.394)$	$7.263 \pm (1.299)$	$7.076 \pm (1.091)$	$6.957 \pm (1.333)$
f_7	$0 \pm (0)$	$0 \pm (0)$	$0 \pm (0)$	$0 \pm (0)$
F	$J_r = 0.09$	$J_r = 0.10$		
f_1	$1.81E-5 \pm (2.43E-5)$	$4.28E-5 \pm (1.22E-4)$		
f_2	$95.743 \pm (0.919)$	$95.664 \pm (0.796)$		
f_3	$1.824E3 \pm (3.576E2)$	$1.857E3 \pm (3.338E2)$		
f_4	$1.795E3 \pm (2.110E2)$	$1.681E3 \pm (1.641E2)$		
f_5	**0.050 ± (0.095)**	$0.072 \pm (0.183)$		
f_6	$6.563 \pm (1.292)$	$6.954 \pm (1.076)$		
f_7	$0 \pm (0)$	$0 \pm (0)$		

TABLE VI

MEAN ± (STANDARD DEVIATION) OF THE BEST FITNESS VALUE OF DE COMPARED TO ODE WITH $J_r = 0.05$, AND CDE_d WITH J_r VALUES FROM 0.01 TO 0.05, IN D=500. THE J_r VALUES USED FOR ODE AND CDE_d ARE BASED ON PARAMETER ANALYSIS RESULTS FOR THE CORRESPONDING ALGORITHMS ACCORDING TO TABLES IV AND V. THE BEST RESULT FOR EACH FUNCTION IS HIGHLIGHTED IN **BOLDFACE**.

	DE	ODE	CDE_d		
F		$J_r = 0.05$	$J_r = 0.01$	$J_r = 0.02$	$J_r = 0.03$
f_1	**0 ± (0)**	$3.52E-6 \pm (4.273E-6)$	$1.096E-5 \pm (3.43E-5)$	$1.80E-5 \pm (6.39E-5)$	$1.33E-4 \pm (6.53E-4)$
f_2	$111.93 \pm (5.75)$	**80.266 ± (5.208)**	$95.103 \pm (1.26)$	$95.089 \pm (0.813)$	$95.278 \pm (0.921)$
f_3	$1828.49 \pm (458.30)$	**1575.803 ± (336.632)**	$1.789E3 \pm (4.429E2)$	$1.809E3 \pm (3.782E2)$	$1.627E3 \pm (2.133E2)$
f_4	$1868.76 \pm (135.15)$	$1608.951 \pm (140.728)$	**1.532E3 ± (1.382E2)**	$1.604E3 \pm (1.062E2)$	$1.635E3 \pm (1.457E2)$
f_5	$0.16 \pm (0.27)$	$0.489 \pm (1.066)$	$0.111 \pm (0.212)$	$0.140 \pm (0.318)$	$0.188 \pm (0.307)$
f_6	$7.62 \pm (1.44)$	$8.436 \pm (1.528)$	$6.660 \pm (1.137)$	**6.464 ± (1.010)**	$6.735 \pm (1.414)$
f_7	$0.06 \pm (0.27)$	$1.784E-3 \pm (2.609E-3)$	**0 ± (0)**	$0 \pm (0)$	$0 \pm (0)$
			$J_r = 0.04$	$J_r = 0.05$	
f_1			$8.72E-6 \pm (1.89E-5)$	$7.12E-6 \pm (1.095E-5)$	
f_2			$95.660 \pm (0.785)$	$95.749 \pm (1.219)$	
f_3			$1.699E3 \pm (3.074E2)$	$1.670E3 \pm (2.262E2)$	
f_4			$1.640E3 \pm (1.449E2)$	$1.630E3 \pm (1.537E2)$	
f_5			$0.067 \pm (0.117)$	**0.051 ± (0.142)**	
f_6			$6.709 \pm (1.042)$	$7.255 \pm (1.394)$	
f_7			$0 \pm (0)$	$0 \pm (0)$	