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

Type-II Opposition-Based Differential Evolution

Hojjat Salehinejad, *Student Member, IEEE,* Shahryar Rahnamayan, *Senior Member, IEEE,* and Hamid R. Tizhoosh, *Member, IEEE*

Abstract— **The concept of opposition-based learning (OBL) can be categorized into Type-I and Type-II OBL methodologies. The Type-I OBL is based on the opposite points in the variable space while the Type-II OBL considers the opposite of function value on the landscape. In the past few years, many research works have been conducted on development of Type-I OBL-based approaches with application in science and engineering, such as opposition-based differential evolution (ODE). However, compared to Type-I OBL, which cannot address a real sense of opposition in term of objective value, the Type-II OBL is capable to discover more meaningful knowledge about problem's landscape. Due to natural difficulty of proposing a Type-II-based approach, very limited research has been reported in that direction. In this paper, for the first time, the concept of Type-II OBL has been investigated in detail in optimization; also it is applied on the DE algorithm as a case study. The proposed algorithm is called opposition-based differential evolution Type-II (ODE-II) algorithm; it is validated on the testbed proposed for the IEEE Congress on Evolutionary Computation 2013 (IEEE CEC-2013) contest with 28 benchmark functions. Simulation results on the benchmark functions demonstrate the effectiveness of the proposed method as the first step for further developments in Type-II OBL-based schemes.**

I. Introduction

With the fast accretion in complexity of real-world problems both in engineering and science, such problems can still be managed using soft computing techniques [1], [2]. The evolutionary optimization algorithms typically generate some initial candidate solutions and then improve them toward an optimum on the landscape, which meets the criteria set by user. The improvement procedure is mostly based on some biological operations. In order to make better decisions by considering the whole variable space as well as landscape in a shorter time, the idea of opposition-based learning (OBL) was introduced in [3], which considers opposite points in the search space.

The OBL concept can be discussed from Type-I and Type-II perspectives [1], [3]. The Type-I OBL deals with the relationship among concepts, based on their attributes, without considering the genuine objective landscape. Generally speaking, the Type-I OBL computation is easy to perform due to its linear definition in the variable space. The Type-I OBL can be considered as an approximation of the genuine intellection of opposite computing, which is the Type-II OBL. The Type-II opposition scheme requires a profound understanding of the objective space. This concept is typically difficult to be utilized in real-world applications, since for the Type-I approach the variable space is already known, while in the Type-II scenario a-priori knowledge about the landscape is required, which is not generally available for black-box problems.

The concept of Type-I OBL has been employed for development of variant well-known machine learning and optimization techniques such as artificial neural networks (ANNs) [4], artificial bee colony (ABC) algorithm [6]-[8], ant colony system (ACS) [9], [11], genetic algorithm (GA) [13], differential evolution (DE) algorithm [14], [15], [18]-[24], fuzzy systems [25], harmony search algorithm (HSA) [26], multi-objective optimization [28], and discrete and combinatorial optimization [29]. In order to solve real-world problems, many Type-I OBL-based algorithms have been utilized such as optimal design of electromagnetic devices [7], large-scale optimization problems with implementation on graphics processing unit (GPU) [30], hydro-thermal scheduling [31], economic load dispatch (ELD) problems of power systems [26], traffic congestion identification [33], placement of radio frequency identification (RFID) readers [34], fuzzy image thresholding [17], [25], breast cancer diagnosis [32], and classification for mass diagnosis in mammography images [27], to mention some among many others. These examples clearly indicate that, Type I OBL has been widely used to develop novel intelligent solutions for practical problems. The idea of OBL can also be utilized in wireless communication systems [12], [16], and vehicular navigation systems [2], for further developments as well.

In this paper, the untouched area of OBL, i.e. Type-II OBL, is investigated and applied to accelerate DE algorithm for the first time. In the literature, there are many opposition-based differential evolution (ODE) inspired algorithms, but all of them similar to the ODE are Type-I based approaches. In other words, for the first time in this paper, the Type-II opposition has been analyzed in depth and utilized in optimization. The Type-II concept makes more sense, since the objective in an optimization problem is finding the optimum value of objective function. In this term, looking for an opposite of value on the objective space can provide deeper knowledge compared to opposite of candidate solution points on the variable space. In this paper, we are going to analyze this challenging concept in detail from different perspectives. Then, the proposed ODE Type-II will be compared with its parents (DE and ODE Type-I) on a well-known test suit with 28 benchmark functions.

Hojjat Salehinejad and Shahryar Rahnamayan 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 (email: {hojjat.salehinejad, shahryar.rahnamayan}@uoit.ca).

Hamid R. Tizhoosh is with the Department of Systems Design Engineering, University of Waterloo, 200 University Avenue West, Waterloo, ON N2L 3G1, Canada (email: tizhoosh@uwaterloo.ca).

In the next section, a brief review of DE algorithm is presented. In the Section III, the concept of OBL is discussed for Type-I and Type-II schemes. The idea of proposed ODE-II algorithm is presented in Section IV and its performance is discussed in details in Section V. Finally, the paper is concluded in Section VI, which includes some further research directions to develop the idea of ODE-II.

II. Differential Evolution

Generally speaking, during solving a black-box optimization problem, an optimizer has no knowledge about the shape of landscape and tries to find optimal decision variables to minimize/maximize an objective function. The DE algorithm is one of the state-of-the-art evolutionary algorithms, which, similar to other algorithms in its category, starts its search procedure with some random initial vectors and tries to improve them generation by generation toward an optimal solution. The population $\mathbf{P} = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$ consists of NP vectors in generation g, where each $\mathbf{X}_i$ is a D-dimensional vector defined as $\mathbf{X}_i = \{x_{i,1}, ..., x_{i,D}\}$. A simple DE algorithm consists of the following three major operations: mutation, crossover, and selection.

Mutation: This step selects three vectors randomly from the population such that $i_1 \neq i_2 \neq i_3 \neq i$ where $i \in \{1, ..., NP\}$ and $NP \geq 4$, for each vector $\mathbf{X}_i$. The mutant vector is calculated as follows:

$$\mathbf{V}_i = \mathbf{X}_{i_1} + F(\mathbf{X}_{i_2} - \mathbf{X}_{i_3}) \tag{1}$$

where the factor $F \in [0, 2]$ is a real constant number, which controls the amplification of the added differential variation of $(\mathbf{X}_{i_2} - \mathbf{X}_{i_3})$. The exploration of DE increases by selecting higher values for F.

Crossover: The crossover operation is to increase diversity of the population by shuffling the mutant and parent vector as follows:

$$U_{i,d} = \begin{cases} V_{i,d}, & rand_d(0,1) \leq Cr \ or \ d_{rand} = d \\ x_{i,d}, & otherwise, \end{cases} \tag{2}$$

where $d = [1, ..., D]$, $Cr \in [0, 1]$ is the crossover rate parameter, and $rand(a, b)$ generates a random number in arbitrary interval $[a, b]$ with uniform distribution. Therefore, the trial vector $\mathbf{U}_i \ \forall i \in \{1, ..., NP\}$ is generated as

$$\mathbf{U}_i = \{U_{i,1}, ..., U_{i,D}\}. \tag{3}$$

Selection: The $\mathbf{U}_i$ and $\mathbf{X}_i$ vectors are evaluated and compared with respect to their fitness values. The one with better fitness is selected for the next generation.

III. Opposition-Based Learning

In this section the concept of OBL is discussed from Type-I and Type-II viewpoints. Then, the min-max-based and centroid-based opposite computing schemes are discussed. Since the OBL was started with the Type-I trend, the min-max oppositional computing has received more attention due to availability of variables' boundaries. However, the centroid opposition computing works based on the centroid point of population and does not require variables' boundaries; which can be employed in Type-II approach.

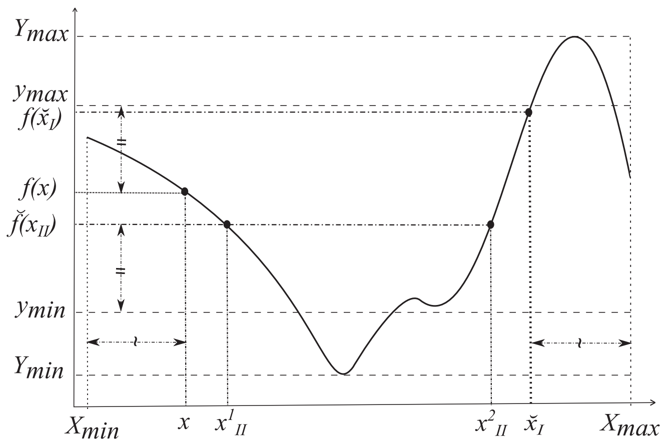


Fig. 1. Type-I versus Type-II opposition definition for a sample landscape.

A. Type-I vs. Type-II Opposition-Based Learning

Typically in population-based algorithms, the idea is generating some random initial candidates and then trying to guide them toward an optimal solution. By employing the concept of considering candidates and corresponding opposite candidates simultaneously, the search algorithm can better explore the landscape via higher search coverage and diversity of its candidate solutions with the chance of having fitter candidates. This idea can be encouraged by the probability theory, where for a specific problem, the opposite of the current candidate solution has a higher chance to be closer to the solution than a random number [22]. This phenomenon can accelerate convergence rate of the optimizer [1], [14]. The concept of OBL can be categorized into Type-I and Type-II schemes [1].

The min-max Type-I OBL can be defined for a D-dimensional point $x_d \in [X_{min,d}, X_{max,d}]$ as

$$\breve{x}_d = X_{min,d} + X_{max,d} - x_d \tag{4}$$

where $d = 1, ..., D$. By considering $\mathbf{X} = (x_1, ..., x_D)$ as a point in the D-dimensional space and $\breve{\mathbf{X}} = (\breve{x}_1, ..., \breve{x}_D)$ as the opposite point of $\mathbf{X}$, if $f(\mathbf{X}) \leq f(\breve{\mathbf{X}})$, where $f(.)$ is the fitness value, $\breve{\mathbf{X}}$ is considered as the better solution in a maximization problem. In order to have a general definition, if $\psi \in \Psi$ is a concept in a D-dimensional space and $\Phi : \Psi \rightarrow \Psi$ is a one-to-one mapping function, which defines an oppositional relationship between two unique elements ψ_1 and ψ_2 of concept class Ψ, the opposition concept for OBL Type-I can be determined as $\breve{\psi} = \Phi(\psi)$. In addition, it is fair to say if $\Phi(\psi_1) = \psi_2$ and then $\Phi(\psi_2) = \psi_1$, the oppositional relationship is symmetric in this case.

In the Type-II OBL, for the point $\mathbf{X}$ we have $f(\mathbf{X}) \in [Y_{min}, Y_{max}]$. Therefore, the min-max opposition computation for Type-II OBL is defined as

$$\breve{f}(\mathbf{X}) = Y_{min} + Y_{max} - f(\mathbf{X}). \tag{5}$$

In practice, since the landscape boundaries are unknown, the boundaries Y_{min} and Y_{max} can be estimated using sampling as y_{min} and y_{max}, respectively. However, it is proved in the next subsection that the centroid-based method

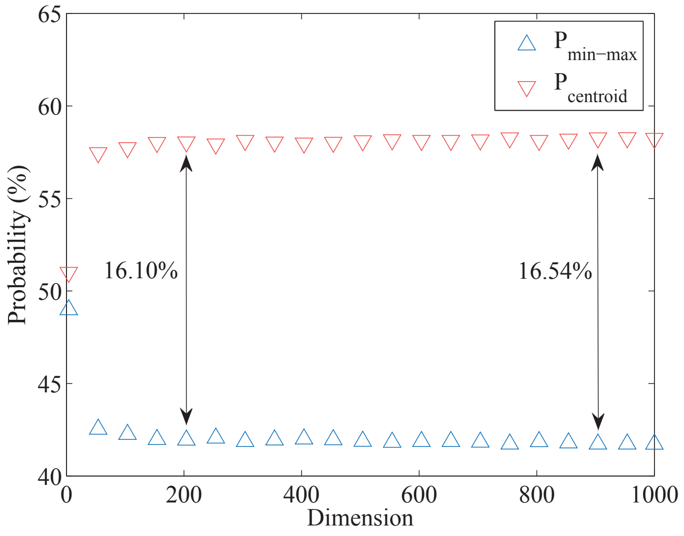


Fig. 2. Monte-Carlo simulation for min-max and centroid oppositional computation.

can demonstrate better estimation compared to min-max-based opposition.

In order to present an example of Type-I and Type-II OBL schemes, a sample nonlinear function for variable range $[X_{min}, X_{max}]$ and landscape range of $[Y_{min}, Y_{max}]$ is illustrated in Fig. 1. In this scenario, an arbitrary point x with its corresponding objective value $f(x)$ is selected. The opposite of x using Type-I approach is calculated using Eq. (4) and denoted by $\breve{x}_I$, where $f(\breve{x}_I)$ demonstrates its corresponding value on the landscape, objective value. By having $f(x)$, its Type-II opposite $\breve{f}(x_{II})$ is calculated using Eq. (5). The corresponding values of $\breve{f}(x_{II})$ in the variable space are denoted on the X-axis as x_{II}^1 and x_{II}^2. As it is demonstrated, for non-monotonic functions Type-II opposition, $\breve{f}(x_{II})$, may refer to more than one values on the variable space, which is referred to one-to-many mapping. In such situation, a simple way is to select one of the one-to-many variable space solutions randomly.

From the above discussion, it is apparent that the Type-II OBL can lead us to more reliable candidate solution than the Type-I OBL approach [1]. Because the objective of an optimization problem is minimizing/maximizing the objective space, so the search should be conducted on the objective space to find the minimum/maximum point(s) and then the corresponding solution(s).

B. Min-Max vs. Centroid-based Oppositional Computation

If we define $f_c(\mathbf{X}) = (Y_{min} + Y_{max})/2$ as the center of interval $[Y_{min}, Y_{max}]$, we can simply replace $(Y_{min}+Y_{max})$ in Eq. (5) with $2f_c(\mathbf{X})$. As it is clear for the Type-II OBL, in a black-box optimization, $f(\mathbf{X})$ as well as its boundaries are not available. In this case, the centroid-based opposition can be employed for calculation of $f(\mathbf{X}_i)$ opposite as

$$\breve{f}(\mathbf{X}_i) = 2f_c(\mathbf{X}) - f(\mathbf{X}_i) \tag{6}$$

where $i = 1, ..., NP$ and $f_c(.)$ is the population centroid defined as follows:

$$f_c(\mathbf{X}) = \frac{1}{NP}\sum_{i=1}^{NP} f(\mathbf{X}_i). \tag{7}$$

We show that by using Monte-Carlo method, in case of not having the 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 as $d = (1, ..., 1000)$ and with $NRun = 1e4$ 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

$$\Delta^r = min\{\delta_1^r, ..., \delta_{NS}^r\}. \tag{8}$$

By considering the min-max and centroid-based opposition calculation approaches for comparison, the distances using both approaches are calculated and then cardinality (the number of runs) in which each one was successful (i.e., closer to optimum) is counted as

$$n_{centroid} = card\{\Delta_{centroid}^r < \Delta_{min-max}^r\} \tag{9}$$

and

$$n_{min-max} = card\{\Delta_{centroid}^r > \Delta_{min-max}^r\}. \tag{10}$$

By having in mind the total number of runs, the success probability for min-max scheme is defined as follow:

$$P_{min-max} = \frac{n_{centroid}}{NRun} \tag{11}$$

and as well for the centroid scheme a similar probability is calculated as follow:

$$P_{centroid} = \frac{n_{min-max}}{NRun} \tag{12}$$

where $P_{centroid} + P_{min-max} = 1$. By using the Eq.s (11) and (12), probability of success for min-max and centroid schemes are illustrated in Fig. 2. It demonstrates an increasing trend for $d < 200$, while for higher dimensions the probability difference among dimensions remains almost flat. Simulation results deliver: $P_{min-max} = 41.95\%$ and $P_{centroid} = 58.05\%$ for $d = 200$ and for $d = 1000$ the $P_{min-max} = 41.73\%$ and $P_{centroid} = 58.27\%$. The results show that the centroid-based oppositional computing not only works for Type-II OBL, as an approach where the landscape boundaries are unknown, but also can outperform the min-max-based oppositional computing approach in term of the probability of closeness to an optimal solution in a black-box problem.

Algorithm 1 ODE-II

```
1: **Procedure** ODE-II
2: Parameter Setting$(Jr, Cr, F, NFC_{Max}, D, NP)$
3: call Algorithm 2: Look-Up Table Generation
4: call Algorithm 3: Population Initialization
5: while $(|BFV > VTR| > EVTR$ & $NFC < NFC_{Max})$ do
6:     call Algorithm 4:Differential Evolution
7:     $g = g + 1$
8:     if $rand(0, 1) \leq Jr$ then
9:         call Algorithm 6: ODE-II Jumping
10:    end if
11: end while
```

Algorithm 2 Look-Up Table Generation

```
1: Procedure Look-Up Table Generation
2: Generate Look-Up Table as $\mathbf{L} = \{\mathbf{X}_1, ..., \mathbf{X}_S\} : \mathbf{X}_i = \{x_{i,1}, ..., x_{i,D}\}, \forall i \in \{1, ..., S\}$
3: Evaluate $\mathbf{L}$ as $EL_i = f(\mathbf{X}_i), \forall i \in \{1, ..., S\}$
```

IV. Type-II Opposition-Based Differential Evolution

It is clear that to utilize Type-II opposition in a black-box optimization problem, having knowledge about the landscape is important. Therefore, inverse-meta-modelling techniques are required to map objective value to the corresponding variables values. In this work, the idea is sampling landscape and using an interpolation method to estimate the variables space values corresponding to objective space values. The initial sampling of landscape is then improved during search process, without paying any additional objective function evaluation cost. In fact, visited extra points during the search process are also added to our sampling set to make it more reliable.

The pusdocode of the proposed ODE-II algorithm is given in Algorithm 1. It consists of different procedures, which are *Look-Up Table Generation* (Algorithm 2), *Population Initialization* (Algorithm 3), *Differential Evolution* (Algorithm 4), and *ODE-II Jumping* (Algorithm 6).

The first step of Algorithm 1 is parameters setting, where Jr is the jumping rate and NFC_{Max} is maximum number of function calls. Then, a sampling of landscape is performed using the *Look-Up Table Generation* procedure as explained in Algorithm 2.

1) Look-Up Table Generation: In this procedure a look-up table $\mathbf{L} = \{\mathbf{X}_1, ..., \mathbf{X}_S\}$ is generated, where $\mathbf{X}_i = \{x_{i,1}, ..., x_{i,D}\}$ for $i = 1, ..., S$. Each space variable is a random number with uniform distribution generated as $x_{i,d} = X_{min} + rand(0, 1) \times (X_{max} - X_{min})$, where $S = NP \times D$ is size of the look-up table, NP is the population size, and D is the problem dimension. The fitness value of samples in $\mathbf{L}$ are denoted by $\mathbf{EL} = f(\mathbf{L})$. Back to Algorithm 1, the first population generation is then constructed as well as selected by the *Population Initialization* procedure.

2) Population Initialization: After *Look-Up Table Generation*, NP population vectors are selected randomly from $\mathbf{L}$ to construct the population $\mathbf{P}$ in the generation $g = 0$ as explained in Algorithm 3. Then, the center point of fitness

Algorithm 3 Population Initialization

```
1: Procedure Population Initialization
2: $g = 0$
3: Select $NP$ random population from $\mathbf{L}$ as $\mathbf{P} = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$
4: $f_c(\mathbf{P}) = \sum_{i=1}^{NP} f(\mathbf{X}_i)/NP, \forall \mathbf{X}_i \in \mathbf{P}$
5: for $i = 1 \to NP$ do
6:     for $d = 1 \to D$ do
7:         $\breve{x}_{I,i,d} = X_{min,d} + X_{max,d} - x_{i,d}$
8:     end for
9:     if $\breve{\mathbf{X}}_{I,i} \in \mathbf{L} : \breve{\mathbf{X}}_{I,i} = \mathbf{X}_k, k \in \{1, ..., S\}$ then
10:        $f(\breve{\mathbf{X}}_{I,i}) = EL_k$
11:    else
12:        Evaluate $\breve{\mathbf{X}}_{I,i}$ as $f(\breve{\mathbf{X}}_{I,i})$
13:        Add $\breve{\mathbf{X}}_{I,i}$ to $\mathbf{L}$
14:        Add $f(\breve{\mathbf{X}}_{I,i})$to $\mathbf{EL}$
15:    end if
16:    $\mathbf{P}_{I,i} = \breve{\mathbf{X}}_{I,i}$
17:    $\breve{f}(\mathbf{X}_{II,i}) = 2f_c(\mathbf{P}) - f(\mathbf{X}_i)$
18:    if $\breve{f}(\mathbf{X}_{II,i}) \in \mathbf{EL} : \breve{f}(\mathbf{X}_{II,i}) = EL_k, k \in \{1, ..., S\}$ then
19:        $\mathbf{X}_{II,i} = \mathbf{X}_k : \mathbf{X}_k \in \mathbf{L}$
20:    else
21:        call Algorithm 5: Interpolation_Centre
22:    end if
23:    $\mathbf{P}_{II,i} = \mathbf{X}_{II,i}$
24: end for
25: Select $NP$ fittest individuals $\mathbf{P}^g = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$ from the set $\{\mathbf{P}, \mathbf{P}_I, \mathbf{P}_{II}\}$
```

values of $\mathbf{P}$ is calculated by Eq. (7).

In order to prepare the Type-I opposite population $\mathbf{P}_I$, opposite of each population member in $\mathbf{P}$ is calculated using the min-max method by Eq. (4) as

$$\breve{x}_{I,i,d} = X_{min,d} + X_{max,d} - x_{i,d}. \tag{13}$$

If objective value of $\breve{\mathbf{X}}_{I,i}$ does not exist in the $\mathbf{EL}$, its objective value is computed as $f(\breve{\mathbf{X}}_{I,i})$. The sampling table $\mathbf{L}$ and its evaluated set $\mathbf{EL}$ is then updated by the new value to develop the look-up table without suffering from an extra objective evaluation cost. The new population vector $\breve{\mathbf{X}}_{I,i}$ is then added to the Type-I opposite set $\mathbf{P}_I$. Since landscape boundaries are unknown, as discussed in subsection B of Section III, the Type-II opposite of $f(\mathbf{X}_i)$ is computed using Eq. (6).

If the fitness value $\breve{f}(\mathbf{X}_{II,i})$ already exists in the $\mathbf{EL}$, $\breve{f}(\mathbf{X}_{II,i}) \in \mathbf{EL}$, the corresponding variables vector $\mathbf{X}_k$ in the $\mathbf{L}$, i.e. $\mathbf{X}_{II,i}$, is picked up and added to the opposite Type-II population $\mathbf{P}_{II}$. Otherwise, an interpolation technique is employed to estimate $\mathbf{X}_{II,i}$. In this paper, the centre-based interpolation technique is employed, which is illustrated in Algorithm 5 and will be discussed later at the end of this section. Finally, the fittest NP individual vectors are selected from the population pool $\{\mathbf{P}, \mathbf{P}_I, \mathbf{P}_{II}\}$ to construct the current population $\mathbf{P}^g = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$.

Back to Algorithm 1, the termination criterion is met when the difference between best fitness value (*BFV*) and fitness value to reach (*VTR*) is less than fitness error value to reach (*EVTR*), or ODE-II searching procedure passes the maximum number of function calls NFC_{Max}, that is

Algorithm 4 Differential Evolution

```
1: Procedure Differential Evolution
2: $\mathbf{P} = \mathbf{P}^g$
3: for $i = 1 \to NP$ do
4:     Select three random population vectors from $\mathbf{P}$ where $(i_1 \neq i_2 \neq i_3 \neq i)$
5:     $\mathbf{V}_i = \mathbf{X}_{i_1} + F(\mathbf{X}_{i_2} - \mathbf{X}_{i_3})$
6:     for $d = 1 \to D$ do
7:         if $rand(0,1) < Cr$ then
8:             $U_{i,d} = V_{i,d}$
9:         else
10:            $U_{i,d} = x_{i,d}$
11:        end if
12:    end for
13:    if $\mathbf{U}_i \in \mathbf{L} : \mathbf{U}_i = \mathbf{X}_k, k \in \{1, ..., S\}$ then
14:        $f(\mathbf{U}_i) = EL_k$
15:    else
16:        Evaluate $\mathbf{U}_i$ as $f(\mathbf{U}_i)$
17:        Add $\mathbf{U}_i$ to $\mathbf{L}$
18:        Add $f(\mathbf{U}_i)$ to $\mathbf{EL}$
19:    end if
20:    if $f(\mathbf{U}_i) \leq f(\mathbf{X}_i)$ then
21:        $\mathbf{X}'_i = \mathbf{U}_i$
22:    else
23:        $\mathbf{X}'_i = \mathbf{X}_i$
24:    end if
25: end for
26: $\mathbf{X}_i = \mathbf{X}'_i, \forall i \in \{1, ..., NP\}$
27: $\mathbf{P}^{g+1} = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$
```

Algorithm 5 Interpolation$_{\text{Centre}}$

```
1: Procedure Interpolation$_{\text{Centre}}$
2: if $\breve{f}(\mathbf{X}_{II,i}) < EL_L$ then
3:     $\breve{f}(\mathbf{X}_{II,i}) = EL_L + (EL_L - \breve{f}(\mathbf{X}_{II,i}))$
4: else
5:     if $\breve{f}(\mathbf{X}_{II,i}) > EL_U$ then
6:         $\breve{f}(\mathbf{X}_{II,i}) = EL_U - (\breve{f}(\mathbf{X}_{II,i}) - EL_U)$
7:     end if
8: end if
9: Find $j$ such as $EL_j < \breve{f}(\mathbf{X}_{II,i}) < EL_{j+1}$ for $j \in \{1, ..., S-1\}$
10: $\mathbf{X}_{II,i} = \dfrac{\mathbf{X}_j + \mathbf{X}_{j+1}}{2} : \{\mathbf{X}_j, \mathbf{X}_{j+1}\} \in \mathbf{L}$
```

$NFC \geq NFC_{Max}$. If the termination criterion is not met, the *Differential Evolution* procedure is run.

3) Differential Evolution: Similar to the *Differential Evolution* procedure described in section II, the *Mutation* and *Crossover* are conducted on the population set $\mathbf{P}$ in Algorithm 4. In order to save the number of $NFCs$ as much as possible, if the trial vector $\mathbf{U}_i$ exists in the $\mathbf{L}$ such as $\mathbf{U}_i \in \mathbf{L}$ for the index k, where $\mathbf{U}_i = \mathbf{X}_k$ and $k \in \{1, ..., S\}$, the corresponding objective value is picked up from the $\mathbf{EL}$ set without suffering from extra objective function computation. Otherwise, fitness value of $\mathbf{U}_i$ is evaluated as $f(\mathbf{U}_i)$. Then, $\mathbf{U}_i$ and its corresponding fitness value are added to the $\mathbf{L}$ and $\mathbf{EL}$ sets, respectively. Following with the *Selection* procedure, the population is obtained as $\mathbf{P}^{g+1} = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$.

Back to Algorithm 1, after updating the generation counter, if $rand(0,1) \leq Jr$, where Jr is the jumping rate, the ODE-II generation jumping will get involved in the in procedure.

Algorithm 6 ODE-II Jumping

```
1: Procedure ODE-II Jumping
2: $\mathbf{P} = \mathbf{P}^g$
3: $f_c(\mathbf{P}) = \sum_{i=1}^{NP} f(\mathbf{X}_i)/NP, \forall \mathbf{X}_i \in \mathbf{P}$
4: for $i = 1 \to NP$ do
5:     for $d = 1 \to D$ do
6:         $x_{I,i,d} = x_{min,d} + x_{max,d} - x_{i,d}$
7:     end for
8:     if $\mathbf{X}_{I,i} \in \mathbf{L} : \mathbf{X}_{I,i} = \mathbf{X}_k, k \in \{1, ..., S\}$ then
9:         $f(\mathbf{X}_{I,i}) = EL_k$
10:    else
11:        Evaluate $\mathbf{X}_{I,i}$ as $f(\mathbf{X}_{I,i})$
12:        Add $\mathbf{X}_{I,i}$ to $\mathbf{L}$
13:        Add $f(\mathbf{X}_{I,i})$to $\mathbf{EL}$
14:    end if
15:    $\mathbf{P}_{I,i} = \mathbf{X}_{I,i}$
16:    $\breve{f}(\mathbf{X}_{II,i}) = 2f_c(\mathbf{P}) - f(\mathbf{X}_i)$
17:    if $\breve{f}(\mathbf{X}_{II,i}) \in \mathbf{EL} : \breve{f}(\mathbf{X}_{II,i}) = EL_k, k \in \{1, ..., S\}$ then
18:        $\mathbf{X}_{II,i} = \mathbf{X}_k$
19:    else
20:        Interpolation$_{\mathbf{M}}$
21:    end if
22:    $\mathbf{P}_{II,i} = \mathbf{X}_{II,i}$
23: end for
24: Select $NP$ fittest individuals $\mathbf{P}^g = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$ from the set $\{\mathbf{P}, \mathbf{P}_I, \mathbf{P}_{II}\}$
```

4) ODE-II Jumping: After obtaining the population $\mathbf{P} = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$, the ODE-II *Jumping* procedure is applied as explained in Algorithm 6. In this procedure, similar to Algorithm 3, the $f_c(\mathbf{P})$ is calculated using Eq. (7) and then the min-max-based opposites of $\mathbf{P}$ are computed using Eq. (4). Since the sampling table $\mathbf{L}$ is regularly updated throughout the ODE-II algorithm, the $\mathbf{L}$ is checked for existence of $\mathbf{X}_{I,i}$'s objective value to avoid having an extra objective function call. Therefore, if $\mathbf{X}_{I,i} \in \mathbf{L}$ such that $\mathbf{X}_{I,i}$ is equal to $\mathbf{X}_k$, where $k \in \{1, ..., S\}$, the index k is detected. The objective value of $\breve{\mathbf{X}}_{I,i}$ is extracted from $\mathbf{EL}$ as $f(\breve{\mathbf{X}}_{I,i})$. In case of not having $\breve{\mathbf{X}}_{I,i}$ in $\mathbf{L}$, its objective value is calculated and added to the $\mathbf{EL}$ set. After updating the $\mathbf{P}_{I,i}$ set, the Type-II opposite point of $\mathbf{X}_i$ is computed using the centroid-based oppositional computing method as in Eq. (6). The corresponding Type-II variable $\mathbf{X}_{II,i}$ is extracted from the look-up table if $\breve{f}(\mathbf{X}_{II,i}) \in \mathbf{EL}$ such that $\breve{f}(\mathbf{X}_{II,i}) = EL_k$ for $k \in \{1, ..., S\}$. Otherwise, the $Interpolation_{Centre}$ method in Algorithm 5 is employed for $\mathbf{X}_{II,i}$ estimation. The final population set is constructed by selecting the NP fittest variable vectors from the $\{\mathbf{P}, \mathbf{P}_I, \mathbf{P}_{II}\}$ set as $\mathbf{P}^g = \{\mathbf{X}_1, ..., \mathbf{X}_{NP}\}$.

5) Interpolation$_{Centre}$: Several interpolation methods can be employed to find an estimate of variable in the Type-II opposition scheme. The center-based interpolation procedure is used in this work as illustrated in Algorithm 5. In this procedure, first the $\breve{f}(\mathbf{X}_{II,i})$ availability is checked to find out whether is it in the range of estimated landscape boundaries of $\mathbf{EL}$ or not, $[EL_L, EL_U]$. This boundary is similar to the one in Fig. 1 presented as $[y_{min}, y_{max}]$. If $\breve{f}(\mathbf{X}_{II,i}) < EL_L$, the $\breve{f}(\mathbf{X}_{II,i})$ is flipped by using the lower limit EL_L as

$$\breve{f}(\mathbf{X}_{II,i}) = EL_L + (EL_L - \breve{f}(\mathbf{X}_{II,i})) \tag{14}$$

and if $\breve{f}(\mathbf{X}_{II,i}) > EL_U$ it is flipped by using the upper limit

$$\breve{f}(\mathbf{X}_{II,i}) = EL_U - (\breve{f}(\mathbf{X}_{II,i}) - EL_U). \tag{15}$$

After boundary check, the lower and upper limits of $\breve{f}(\mathbf{X}_{II,i})$ are looked up in the **EL** such that

$$EL_j < \breve{f}(\mathbf{X}_{II,i}) < EL_{j+1} \tag{16}$$

for $j \in \{1, ..., S-1\}$to find the index j. By using the center-based interpolation, we can obtain an estimated vector $\mathbf{X}_{II,i}$ as

$$\mathbf{X}_{II,i} = \frac{\mathbf{X}_j + \mathbf{X}_{j+1}}{2} \tag{17}$$

where $\{\mathbf{X}_j, \mathbf{X}_{j+1}\} \in \mathbf{L}$.

V. Simulations

In this section, the proposed ODE-II algorithm is tested and compared with DE and ODE algorithms. In the next subsection, the parameters setting as well as employed benchmark functions are explained. Then, the comparison strategies and metrics for performance evaluations are presented. Later, the detailed simulations and visualizations are conducted.

A. Parameter Setting and Benchmark Functions

All the experiments have been conducted on the CEC-2013 testbed [10], which is an improved version of CEC-2005 [5] counterpart with additional test functions and modified formula of the composition functions, oscillations, and symmetric-breaking transforms. This testbed contains 28 benchmark functions, divided into three categories which are uni-modal functions ($f_1 - f_5$), multi-modal functions ($f_6 - f_{20}$), and composition functions ($f_{21} - f_{28}$) [5], [10]. Parameters setting for all the experiments are presented in Table I adapted from the literature unless a change is mentioned [10], [14]. The reported values are averaged for 50 independent runs per function.

B. Comparison Strategy and Metrics

In order to have a measure of convergence speed between two algorithms Alg_1 and Alg_2, the acceleration rate (AR) is considered as

$$AR = \frac{NFCs_{Alg_1}}{NFCs_{Alg_2}} \tag{18}$$

where $NFCs_{Alg_1}$ and $NFCs_{Alg_2}$ represent $NFCs$ of Alg_1 and Alg_2, respectively. Therefore, if $AR > 1$ it means that Alg_2 is faster than Alg_1. The number of times an algorithm reaches the optimal solution, i.e. $|BFV - VTR| \leq EVTR$ for $NFC < NFC_{Max}$, is reported based on the success rate (*SR*) measure defined by

$$SR = \frac{card\{|BFV_r - VTR| \leq EVTR\}}{NRun}, \tag{19}$$

for all $r \in \{1, ..., NRun\}$.

Since both SR and NFC are important measures to compare performance of algorithms in terms of convergence rate and robustness, these measures are combined together as a measure called success performance (SP) [5], [14], which is defined as follow:

$$SP = \frac{NFCs}{SR}. \tag{20}$$

The lower SP means a higher performance. Therefore, SP_N is defined as the number of times in percentage that an algorithm has the best SP among other competing algorithms on the 28 benchmark functions.

TABLE I

Parameter setting for all the experiments

Parameter	Description	Value
Jr	Jumping Rate Constant	0.3
NP	Population Size	50
F	Differential Amplification Factor	0.5
Cr	Crossover Probability Constant	0.9
NFC_{Max}	Maximum Number of Function Calls	$1e4 \times D$
$EVTR$	Objective Function Error Value to Reach	1e-8
$NRun$	Number of Runs	50

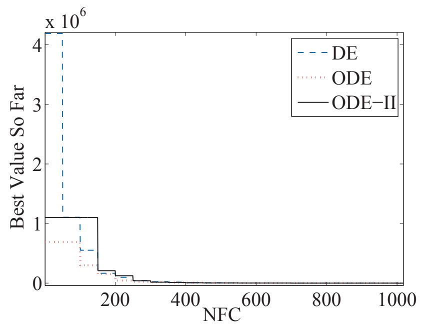
TABLE II

Performance comparison of DE, ODE, and ODEII on CEC-2013 benchmark functions. The highest success performance SP is highlighted in boldface.

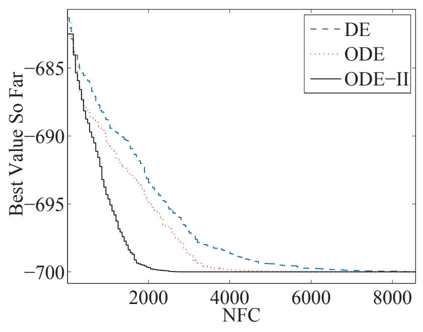
F	DE		ODE		ODE-II			
	NFC	SR	NFC	SR	NFC	SR	AR_1	AR_2
1	2438	100	2180	100	**1869**	**100**	1.30	1.17
2	3737	100	3453	100	**2945**	**100**	1.27	1.17
3	**4477**	**100**	4549	100	4985	94	0.90	0.91
4	3702	100	3537	100	**2896**	**100**	1.28	1.22
5	3222	100	2883	100	**2372**	**100**	1.36	1.22
6	2534	100	3111	100	**2427**	**100**	1.04	1.28
7	7450	100	8453	100	**5589**	**98**	1.33	1.51
8	7677	100	7516	100	**4779**	**98**	1.61	1.57
9	7904	100	8735	100	**5041**	**100**	1.57	1.73
10	6224	100	5524	100	**4424**	**100**	1.41	1.25
11	3598	100	3408	100	**2789**	**100**	1.29	1.22
12	4217	100	4218	100	**3066**	**100**	1.38	1.38
13	3833	100	3789	100	**2831**	**100**	1.35	1.34
14	8364	84	7896	90	**6558**	**92**	1.28	1.20
15	13311	54	**11777**	**64**	13489	44	0.99	0.87
16	**18678**	**94**	19403	28	19506	20	0.96	0.99
17	10731	72	**10369**	**66**	11419	58	0.94	0.91
18	**14076**	**54**	14778	54	14993	42	0.94	0.99
19	**3434**	**100**	3901	100	3550	98	0.97	1.10
20	14241	52	**12602**	**64**	14590	38	0.98	0.86
21	5313	100	5129	100	**3636**	**100**	1.46	1.41
22	7112	100	7349	100	**5377**	**98**	1.32	1.37
23	7490	100	7743	100	**5280**	**98**	1.42	1.47
24	6957	96	7151	94	**5091**	**94**	1.37	1.40
25	9584	76	**7176**	**92**	10937	56	0.88	0.66
26	7930	86	8383	84	**7545**	**80**	1.05	1.11
27	16863	34	**14610**	**56**	17810	16	0.95	0.82
28	6352	94	6593	90	**4735**	**94**	1.34	1.39

C. Results Analysis

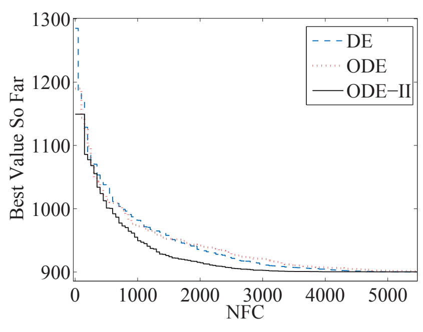
1) Comparison of DE, ODE, and ODE-II Algorithms: The performances of DE, ODE, and ODE-II algorithms are compared by using NFC, SR, and AR values in Table II for $D = 2$. Based on Eq. (18), the terms AR_1 and AR_2 refer to NFC_{DE}/NFC_{ODE-II} and NFC_{ODE}/NFC_{ODE-II}, respectively. The results regarding AR metric show that the



(a) Benchmark function f_4.



(b) Benchmark function f_8.



(c) Benchmark function f_{23}.

Fig. 3. Best value so far (BFV) versus number of function calls (NFC) performance graphs as an example of performance comparison among DE, ODE, and ODE-II algorithms.

TABLE III
THE SUMMARIZED PERFORMANCE RESULTS OF DE, ODE, AND ODE-II FOR SR_{Avg}, NFC_{Avg}, AND SP_N COMPARISON METRICS.

Metric	DE	ODE	ODE-II
SP_N	%21.42	%14.28	**%64.28**
SR_{Avg}	**89.14**	83.79	82.79
NFC_{Avg}	7551.75	7364.86	**6804.61**

ODE-II has outperformed the DE and ODE algorithms on $\%67.86$ and $\%71.43$ of benchmark functions, respectively.

In Table II, the results of algorithm with the highest SP for each benchmark function is highlighted in boldface. It shows that the ODE-II has outperformed the DE and ODE in term of the best SPs. In Table III, the results regarding SP_N metric show that ODE-II has succeeded $\%64.28$ of benchmark functions, while the SP_N for DE and ODE algorithms are $\%21.42$ and $\%14.28$, respectively. Average of $NFCs$ and SRs are denoted by SR_{Avg} and $NFCs_{Avg}$ as in Table III. Regarding SR_{Avg}, the DE algorithm has a higher average SR. In term of $NFCs_{Avg}$ the ODE-II has outperformed the DE and ODE algorithms.

The functions numbers f_4, f_8, and f_{23} from the uni-modal, multi-modal, and composition functions classes are selected for detailed performance analysis, respectively as in Fig. 3. According to our observations, the ODE-II has a sharper move toward convergence to the global optimal solution in the exploration phase than the rest of algorithms. This is while the DE algorithm is more effective during exploitation phase. This fact is discussed in detail by considering participation of DE-based, Type-I-based, and Type-II-based individuals in the population after each generation jumping.

2) Contribution of Type-I and Type-II Opposite Individuals after each Jumping: The term contribution refers to portion of population each scheme Type-I, Type-II, and DE, has in the population. As it was discussed in the Section IV, the DE and ODE algorithms have contribution in making the population after each generation jumping of ODE-II algorithm. Fig. 4 illustrates the contribution (i.e., number of the individuals) of Type-I and Type-II oppositions after each generation jumping. The ODE-II has the most contribution

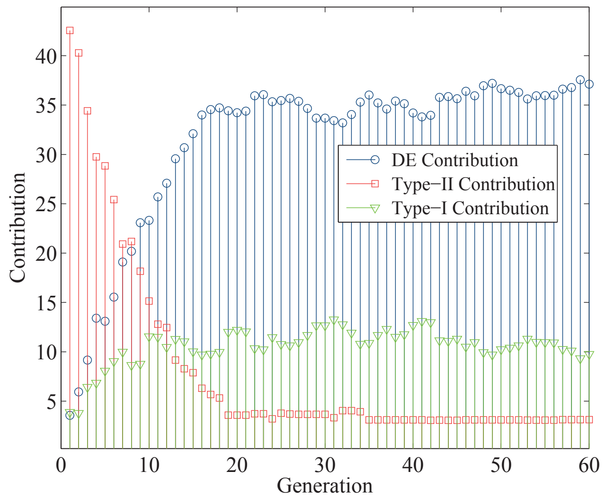


Fig. 4. Contribution (i.e., number of individuals after the jumping) of DE, Type-I OBL, and Type-II OBL in each population generation for benchmark function f_8; the optimal fitness value is -700; 50 runs conducted for each generation.

in the early generations during the exploration phase. As the number of generations increases, its contribution decreases during the exploitation phase. The fine tuning on the other side is mostly done by the DE population. The DE has a few contribution in the exploration phase and as the search moves forward, its contribution in the population increases to have more fine tuning. The Type-I opposition has a few more contributions in exploitation phase than the exploration. However, totally it has less contribution than both DE and Type-II.

VI. CONCLUSION REMARKS

The opposition-based learning (OBL) considers both candidate solution and its opposite to increase the diversity in the search process. So far the OBL concept has been categorized into Type-I and Type-II schemes, in which the Type-I looks for opposition in variable space, while the Type-II considers objective values. Many works have been conducted on the Type-I OBL, however, the concept of opposition thinking is matched highly with Type-II opposition and can offer more knowledge to the optimizer. In this paper, the concept of Type-II OBL has been investigated in detail for the first time to overcome its difficulty to be utilized in a black-box optimization problem. As a case study, we have utilized it for accelerating differential evolution (DE) algorithm. This work can be considered as a preliminary attempt in employing Type-II OBL concept in optimization. There are many interesting but challenging directions in this field to embark upon, such as proposing a fast and accurate method for large-scale inverse meta-modeling and managing the cooperation between ODE-II components.

REFERENCES

[1] H. R. Tizhoosh, M. Ventresca, and S. Rahnamayan, "Opposition-Based Computing," in *Oppositional Concepts in Computational Intelligence*, H. R. Tizhoosh and M. Ventresca, Eds. Springer Berlin Heidelberg, 2008, pp. 11-28.

[2] 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.

[3] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Int. Conf. Comput. Intell. Model. Control Autom. Int. Conf. Intell. Agents, Web Technol. Internet Commer.*, vol. 1, 2005, pp. 695-701.

[4] M. Ventresca and H. R. Tizhoosh, "Improving gradient-based learning algorithms for large scale feedforward networks," in *International Joint Conference on Neural Networks*, 2009, pp. 3212-3219.

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

[6] M. El-Abd, "Generalized opposition-based artificial bee colony algorithm," in *IEEE Congress on Evolutionary Computation,* 2012, pp. 1-4.

[7] X. Zhang, X. Zhang, S. Y. Yuen, S. L. Ho, and W. N. Fu, "An improved artificial bee colony algorithm for optimal design of electromagnetic devices," *IEEE Trans. Magn.*, vol. 49, no. 8, pp. 4811-4816, 2013.

[8] H. Sharma, J. C. Bansal, and K. V. Arya, "Opposition based levy flight artificial bee colony," *Memetic Comput.*, vol. 5, no. 3, pp. 213-227, Dec. 2012.

[9] 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.

[10] J. J. Liang, B.-Y. Qu, P. N. Suganthan, and A. G. Hernandez-Dfaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization," Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Tech. Rep., No. 201212, 2013.

[11] F. Pouladi, A. M. Gilani, B. Nikpour, and H. Salehinejad, "Optimum localization of wind turbine farms using opposition based ant colony optimization," in *International Conference on Developments in E-systems Engineering*, 2013.

[12] H. Salehinejad and S. Talebi, "PAPR Reduction of OFDM Signals by Novel Global Harmony Search in PTS Scheme," *International Journal of Digital Multimedia Broadcasting*, vol. 2012, Article ID 940849, 7 pages, 2012.

[13] 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.

[14] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-Based Differential Evolution," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 64-79, Feb. 2008.

[15] X. Zhang and S. Y. Yuen, "Opposition-based adaptive differential evolution," in *IEEE Congress on Evolutionary Computation*, 2012, pp. 18.

[16] 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.

[17] H. R. Tizhoosh, "Opposite fuzzy sets with applications in image processing," in *Joint International Fuzzy Systems Association World Congress and European Society of Fuzzy Logic and Technology Conference*, 2009, pp. 36-41.

[18] H. Wang, Z. Wu, S. Rahnamayan, and J. Wang, "Diversity analysis of opposition-based differential evolution - An experimental study," in *Advances in Computation and Intelligence*, Z. Cai, C. Hu, Z. Kang, and Y. Liu, Eds. Springer Berlin Heidelberg, 2010, pp. 95-102.

[19] S. Rahnamayan, H. R. Tizhoosh, and M. M. a. Salama, "Quasi-oppositional differential evolution," in *IEEE Congress on Evolutionary Computation*, 2007, no. 1, pp. 2229-2236.

[20] S. Rahnamayan and H. R. Tizhoosh, "Differential Evolution Via Exploiting Opposite," in *Oppositional Concepts in Computational Intelligence*, H. R. Tizhoosh and M. Ventresca, Eds. Springer Berlin Heidelberg, 2008, pp. 143-160.

[21] L. Peng and Y. Wang, "Differential evolution using uniform-quasi-opposition for initializing the population," *Inf. Technol. J.*, vol. 9, no. 8, pp. 1629-1634, 2010.

[22] A. Esmailzadeh and S. Rahnamayan, "Opposition-Based Differential Evolution with Protective Generation Jumping," in *IEEE Symposium on Differential Evolution*, 2011, pp. 1-8.

[23] M. A. Ahandani and H. Alavi-Rad, "Opposition-based learning in the shuffled differential evolution algorithm," *Soft Comput.*, vol. 16, no. 8, pp. 1303-1337, Feb. 2012.

[24] A. Esmailzadeh and S. Rahnamayan,"Opposition-Based Differential Evolution with Protective Generation Jumping," in *IEEE Symposium on Differential Evolution*, 2011, pp. 1-8.

[25] F. S. Al-Qunaieer, H. R. Tizhoosh, and S. Rahnamayan, "Oppositional fuzzy image thresholding," in *International Conference on Fuzzy Systems*, 2010, pp. 1-7.

[26] A. Chatterjee, S. P. Ghoshal, and V. Mukherjee, "Solution of combined economic and emission dispatch problems of power systems by an opposition-based harmony search algorithm," *Int. J. Electr. Power Energy Syst.*, vol. 39, no. 1, pp. 9-20, Jul. 2012.

[27] F. Saki, A. Tahmasbi, and S. B. Shokouhi, "A novel opposition-based classifier for mass diagnosis in mammography images," in *17th Iranian Conference of Biomedical Engineering*, 2010, pp. 1-4.

[28] N. Dong and Y. Wang, "Multiobjective differential evolution based on opposite operation," in *International Conference on Computational Intelligence and Security*, 2009, pp. 123-127.

[29] M. Ergezer, S. Member, D. Simon, and S. Member, "Oppositional biogeography-based optimization for combinatorial problems," in *IEEE Congress on Evolutionary Computation*, 2011, pp. 1496-1503.

[30] X. Zhou, Z. Wu, and H. Wang, "Elite opposition-based differential evolution for solving large-scale optimization problems and its implementation on GPU," in *13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2012, pp. 727-732.

[31] P. Kumar Roy, A. Sur, and D. K. Pradhan, "Optimal short-term hydro-thermal scheduling using quasi-oppositional teaching learning based optimization," *Eng. Appl. Artif. Intell.*, vol. 26, no. 10, pp. 2516-2524, 2013.

[32] F. Saki, A. Tahmasbi, H. Soltanian-Zadeh, and S. B. Shokouhi, "Fast opposite weight learning rules with application in breast cancer diagnosis," *Comput. Biol. Med.*, vol. 43, no. 1, pp. 32-41, 2013.

[33] Y. Yang, Z. Cui, J. Wu, G. Zhang, and X. Xian, "Fuzzy c-means clustering and oposition-based reinforcement learning for traffic congestion identification fuzzy c-means (FCM) clustering clgorithm," *J. Inf. Comput. Sci.*, vol. 9, pp. 2441-2450, 2012.

[34] Y. Gao, X. Hu, H. Liu, F. Li, and L. Peng, "Opposition-based learning estimation of distribution algorithm with gaussian copulas and its application to placement of RFID readers," in *Artificial Intelligence and Computational Intelligence*, F. Deng, Hepu and Miao, Duoqian and Lei, Jingsheng and Wang, Ed. Springer Berlin Heidelberg, 2011, pp. 219-227.