Available online at www.sciencedirect.com
ScienceDirect
Applied Soft Computing 8 (2008) 906–918
Applied Soft Computing
www.elsevier.com/locate/asoc

Opposition versus randomness in soft computing techniques

Shahryar Rahnamayan, Hamid R. Tizhoosh [*], Magdy M.A. Salama

Faculty of Engineering, University of Waterloo, Waterloo, Canada

Received 18 October 2006; received in revised form 30 March 2007; accepted 28 July 2007
Available online 6 August 2007

Abstract

For many soft computing methods, we need to generate random numbers to use either as initial estimates or during the learning and search process. Recently, results for evolutionary algorithms, reinforcement learning and neural networks have been reported which indicate that the simultaneous consideration of randomness and opposition is more advantageous than pure randomness. This new scheme, called *opposition-based learning*, has the apparent effect of accelerating soft computing algorithms. This paper mathematically and also experimentally proves this advantage and, as an application, applies that to accelerate differential evolution (DE). By taking advantage of random numbers and their opposites, the optimization, search or learning process in many soft computing techniques can be accelerated when there is no a priori knowledge about the solution. The mathematical proofs and the results of conducted experiments confirm each other.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Soft computing; Opposition-based learning; Random numbers; Opposite numbers; Differential evolution

1. Introduction

The footprints of the opposition concept can be observed in many areas around us. This concept has sometimes been labeled by different names. Opposite particles in physics, antonyms in languages, complement of an event in probability, antithetic variables in simulation, opposite proverbs in culture, absolute or relative complement in set theory, subject and object in philosophy of science, good and evil in animism, opposition parties in politics, theses and antitheses in dialectic, opposition day in parliaments, and dualism in religions and philosophies are just some examples to mention. Table 1 contains more instances and corresponding details.

It seems that without using the opposition concept, the explanation of different entities around us is hard and maybe even impossible. In order to explain an entity or a situation we sometimes explain its opposite instead. In fact, opposition often manifests itself in a balance between completely different entities. For instance, the east, west, south, and north can not be defined alone. The same is valid for cold and hot and many other examples. Extreme opposites constitute our upper and lower boundaries. Imagination of the infinity is vague, but when we consider the limited, it then becomes more imaginable because its opposite is definable.

Many machine intelligence or soft computing algorithms are inspired by different natural systems. Genetic algorithms, neural nets, reinforcement agents, and ant colonies are, to mention some examples, well established methodologies motivated by evolution, human nervous system, psychology and animal intelligence, respectively. The learning in natural contexts such as these is generally sluggish. Genetic changes, for instance, take generations to introduce a new direction in the biological development. Behavior adjustment based on evaluative feedback, such as reward and punishment, needs prolonged learning time as well.

In many cases the learning begins at a random point. We, so to speak, begin from scratch and move toward an existing solution. The weights of a neural network are initialized randomly, the population initialization in evolutionary algorithms (e.g., GA, DE, and PSO) is performed randomly, and the action policy of reinforcement agents is initially based on randomness, to mention some examples.

Generally, we deal with complex control problems [1,2]. The random guess, if not far away from the optimal solution, can result in a fast convergence. However, it is natural to state that if we begin with a random guess, which is very far away

[*] Corresponding author.
E-mail addresses: shahryar@pami.uwaterloo.ca (S. Rahnamayan), tizhoosh@uwaterloo.ca (H.R. Tizhoosh), m.salama@ece.uwaterloo.ca (M.M.A. Salama).

1568-4946/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2007.07.010

Table 1
Footprints of opposition in different fields

Example	Field	Description
Opposite particles/elements	Physics	Such as magnetic poles (N and S), opposite polarities (+ and −), electron-proton in an atom, action-reaction forces in Newton's third law, and so on.
Antonyms	Language	A word that means the opposite of another word (e.g., hot/cold, fast/slow, top/down, left/right, day/night, on/off).
Antithetic variables	Simulation	Antithetic (negatively correlated) pair of random variables used for variance reduction.
Opposite proverbs	Culture	Two proverbs with the opposite advice or meaning (e.g., the pen is mightier than the sword. Actions speak louder than words.); proverb or its opposite pair offers an applicable solution based on specific situation or condition.
Complements	Set theory	(a) Relative complement ($B - A = \{x \in B \mid x \notin A\}$), (b) absolute complement ($A^c = U - A$, where U is the universal set).
Opposition	Politics	A political party or organized group opposed to the government.
Inverter	Digital design	Output of the inverter gate is one if input is zero and vice versa.
Opposition day	Legislation	A day in the parliament in which small opposition parties are allowed to propose the subject for debate (e.g., Canada's parliament has 25 opposition days).
Dualism	Philosophy and religion	Two fundamental principles/concepts, often in opposition to each other; such as "Yin" and "Yang" in Chinese philosophy and Taoist religion, "subject" and "object" in philosophy of science, "good" and "evil" in animism, "ahura" and "ahriman" in Zarathustra.
Dialectic	Philosophy	An exchange of "theses" and "antitheses" resulting in a "synthesis" (e.g., in Hinduism, these three elements are creation (Brahma), maintenance of order (Vishnu) and destruction or disorder (Shiva)).
Classical elements	Archetype	A set of archetypal classical elements to explain patterns in nature (e.g., the Greek classical elements are Fire (hot and dry), Earth (cold and dry), Air (hot and wet), and Water (cold and wet)).
If-then-else	Algorithm	if *condition* then *statements* [else *elsestatements*], the *else statements* are executed when the opposite of the *condition* happens.
Complement of an event	Probability	$P(A') = 1 - P(A)$.
Revolution	Socio-political	A significant socio-political change in a short period of time.

from the existing solution, let say in worst case it is in the *opposite location*, then the approximation, search or optimization will take considerably more time, or in worst case becomes intractable. Of course, in absence of any a priori knowledge, it is not possible to make the best initial guess. Logically, we should be looking in all directions simultaneously, or more efficiently, in the opposite direction.

The scheme of opposition-based learning (OBL) was introduced by H.R. Tizhoosh [3]. In a very short time, it was applied to enhance reinforcement learning [4–6,8,18,34], differential evolution [10–14], backpropagation neural networks [15,17], simulated annealing [9], ant colony optimization [16], and window memorization for morphological algorithms [7]. Although, the achieved empirical results in the mentioned papers confirm that the concept of opposition-based learning is general enough and can be utilized in a wide range of learning and optimization fields to make these algorithms faster. We are still faced with this fundamental question: Why an opposite number is beneficial compared to an independent random number? In other words, why a second random number should not be used instead of an opposite number? Many optimization methods need to generate random numbers for initialization or using them during the search process, such as differential evolution (DE) [36], genetic algorithms (GAs) [20], ant colony optimization (ACO) [21], particle swarm optimization (PSO) [22], simulated annealing (SA) [23], and random search (RS) [24].

In this paper, we will prove that, in terms of convergence speed, utilizing random numbers and their opposites is more beneficial than using the pure randomness to generate initial estimates in absence of a priori knowledge about the solution. In fact, we mathematically and empirically show why opposite numbers have higher chance to be a fitter estimate compared to additional independent random guesses.

Finally, a population-based algorithm, differential evolution (DE), is selected to be accelerated using the opposition concept. The main reason for selecting DE is its high ability to find precise solutions for the mixed-typed black-box global optimization problems [25]. The population-based algorithms are computationally expensive and hence their acceleration is widely appreciated. Among the various evolutionary algorithms [26–28], differential evolution (DE) is well known for its effectiveness and robustness. Frequently reported studies demonstrate that the DE outperforms many other optimizers over both benchmark functions and real-world applications.

The rest of this paper is organized as follows: Section 2 covers the definitions, theorems, and proofs corresponding to opposition-based learning. Empirical verification of the mathematical results are given in Section 3. Employing of OBL to accelerate differential evolution is investigated in Section 4. Finally, the paper is concluded in Section 5.

2. Opposition-based learning (OBL): Definitions, theorems, and proofs

Definition 1. Let x be a real number in an interval $[a, b]$ $(x \in [a, b])$; the opposite of x, denoted by $\breve{x}$, is defined by

$$\breve{x} = a + b - x. \tag{1}$$

Fig. 1 (top) illustrates x and its opposite $\breve{x}$ in interval $[a, b]$. As seen, x and $\breve{x}$ are located in equal distance from the interval's center ($|(a + b)/2 - x| = |\breve{x} - (a + b)/2|$) and the interval's boundaries ($|x - a| = |b - \breve{x}|$) as well.

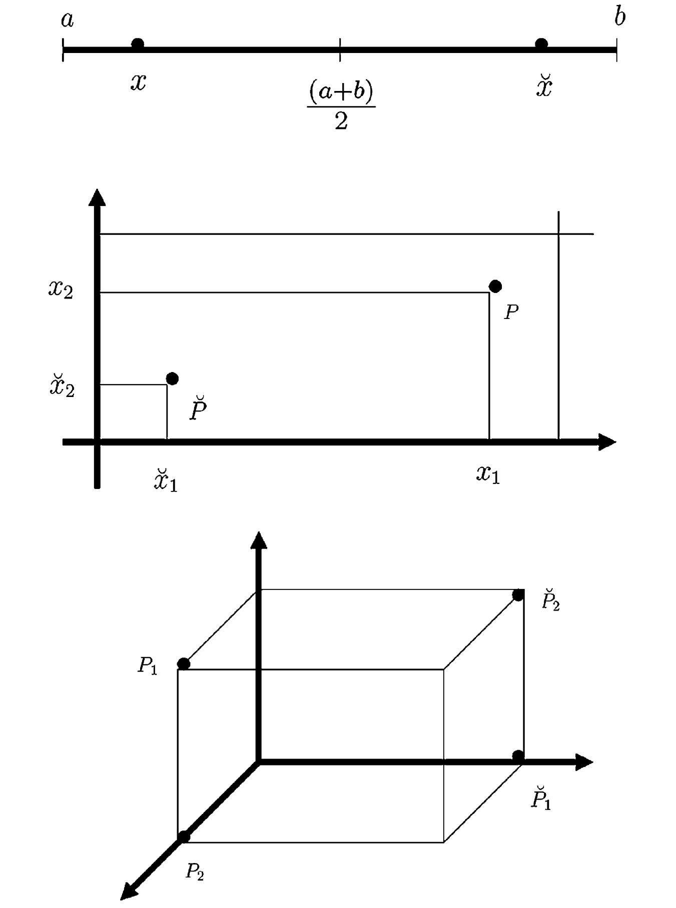

Fig. 1. Illustration of a point and its corresponding opposite in one-, two-, and three-dimensional spaces.

This definition can be extended to higher dimensions by applying the same formula to each dimension [3].

Definition 2. Let $P(x_1, x_2, \ldots, x_D)$ be a point in D-dimensional space, where $x_1, x_2, \ldots, x_D$ are real numbers and $x_i \in [a_i, b_i]$, $i = 1, 2, \ldots, D$. The opposite point of P is denoted by $\breve{P}(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ where

$$\breve{x}_i = a_i + b_i - x_i. \quad (2)$$

Fig. 1 illustrates a sample point and its corresponding opposite point in one-, two-, and three-dimensional spaces.

Theorem 1. Uniqueness *Every point $P(x_1, x_2, \ldots, x_D)$ in the D-dimensional space of real numbers with $x_i \in [a_i, b_i]$ has a unique opposite point $\breve{P}(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ defined by $\breve{x}_i = a_i + b_i - x_i$, $i = 1, 2, 3, \ldots, D$.*

Proof. Let us consider, without lose of generality, the two space corners a_1 and b_1 for the one-dimensional case. According to the opposite point definition, we have $|x - a_1| = |\breve{x} - b_1|$ or $|\breve{x} - a_1| = |x - b_1|$. Now, assume that a second point x' is also opposite of x. Then, we should have $|x - a_1| = |x' - b_1|$ or $|x' - a_1| = |x - b_1|$. This, however, means $x' = \breve{x}$. Hence, $\breve{x}$ is unique. By this way, the uniqueness is supported for each dimension of $\breve{P}(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ regarding $P(x_1, x_2, \ldots, x_D)$, so, $\breve{P}$ is unique as well.

Now, after the definition of the opposite points we are ready to define Opposition-based learning (OBL). □

Definition 3. Let $P(x_1, x_2, \ldots, x_D)$, a point in a D-dimensional space with $x_i \in [a_i, b_i]$ $(i = 1, 2, 3, \ldots, D)$, be a candidate solution. Assume $f(x)$ is a fitness function which is used to measure candidate optimality. According to opposite point definition, $\breve{P}(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ is the opposite of $P(x_1, x_2, \ldots, x_D)$. Now, if $f(\breve{P}) \geq f(P)$, then point P can be replaced with $\breve{P}$; otherwise we continue with P. Hence, the point and its opposite point are evaluated simultaneously to continue with the fitter one [3].

This definition of OBL is making two fundamental assumptions. First, one of the candidate or the opposite candidate is always closer to the solution or are in the same distance from that, and second, considering the opposition is more beneficial than generating additional random solutions and taking the best among them.

Empirical evidence for these claims will be provided in Section 3. However, prior to providing experimental results, we also want to provide mathematical proofs.

Definition 4. Euclidean distance between two points $P(p_1, p_2, \ldots, p_D)$ and $Q(q_1, q_2, \ldots, q_D)$ in a D-dimensional space is defined by

$$\mathrm{d}(P, Q) = \|P, Q\| = \sqrt{\sum_{i=1}^{D} (p_i - q_i)^2}. \quad (3)$$

It can be simplified as follows for a one-dimensional space:

$$\mathrm{d}(P, Q) = \|P, Q\| = |p - q|. \quad (4)$$

Theorem 2. First opposition theorem *For any (unknown) function $y = f(x)(x \in [a, b])$ with global optimum at $x_s(x_s \neq (a+b)/2)$, the estimate solution x and its opposite $\breve{x}$, we have*

$$Pr(|\breve{x} - x_s| < |x - x_s|) = \frac{1}{2}, \quad (5)$$

where $Pr(\cdot)$ is the probability function. It means candidate solution and its opposite have the equal chance to be closer to the solution.

Proof. $x > (a+b)/2 \Leftrightarrow \breve{x} \in [a, (a+b)/2]$ and $x < (a+b)/2 \Leftrightarrow \breve{x} \in [(a+b)/2, b]$. If $x_s \neq (a+b)/2$, then $Pr(|\breve{x} - x_s| < |x - x_s|) = 1/2$. For $x_s = (a+b)/2$, then $Pr(|\breve{x} - x_s| = |x - x_s|) = 1$. □

Now, lets focus on the second assumption of OBL. Suppose the random variables $x_1, x_2, \ldots$ are continuous independent random variables representing the system inputs. Suppose the performance of the system for given inputs x_i is a monotone function $g(x_i)$. We wish to compare the performance of a system with independent inputs with one using opposition-based inputs. In particular, we wish to maximize some measure of performance $g(x)$ over possible inputs x. The following theorem shows the benefit of the opposite inputs, compared to random inputs.

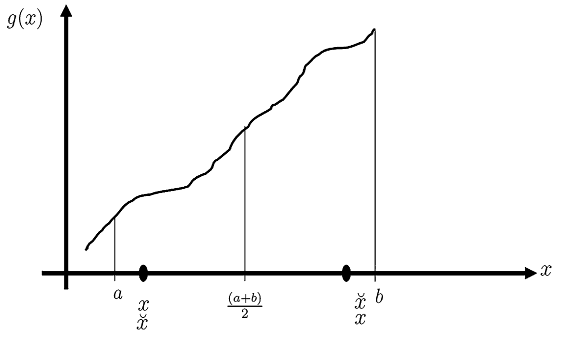

Fig. 2. Illustration of increasingly monotone g, interval boundaries, candidate and opposite candidate solutions.

Theorem 3. Second opposition theorem *For increasingly monotone* g, $Pr(g(x_r) < \max\{g(x), g(\breve{x})\}) = 3/4$, *where* x *is the first random guess*; $\breve{x}$ *is the opposite point of* x; *and* x_r *is the second random guess.*

Proof. Let us prove $Pr(g(x_r) \geq \max\{g(x), g(\breve{x})\}) = 1 - Pr(g(x_r) < \max\{g(x), g(\breve{x})\}) = 1/4$ instead (see Fig. 2),

$$\begin{aligned}
&Pr(g(x_r) > \max\{g(x), g(\breve{x})\}) \\
&= Pr\left(x < \frac{a+b}{2}\right) \times Pr\left(\breve{x} > \frac{a+b}{2}\right) \times Pr\left(x_r > \frac{a+b}{2}\right) \\
&\quad \times Pr(x_r > \breve{x}) + Pr\left(x > \frac{a+b}{2}\right) \times Pr\left(\breve{x} < \frac{a+b}{2}\right) \\
&\quad \times Pr\left(x_r > \frac{a+b}{2}\right) \times Pr(x_r > x) \\
&= \frac{1}{2} \times 1 \times \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times 1 \times \frac{1}{2} \times \frac{1}{2} = \frac{1}{8} + \frac{1}{8} = \frac{1}{4}.
\end{aligned}$$

Following results are obtained from this theorem:

1. $Pr(g(x) > g(x_r)) = Pr(g(x_r) < \max\{g(x), g(\breve{x})\}) \times \frac{1}{2} = \frac{3}{4} \times \frac{1}{2} = \frac{3}{8} = 0.375$,
2. $Pr(g(\breve{x}) > g(x_r)) = Pr(g(x_r) < \max\{g(x), g(\breve{x})\}) \times \frac{1}{2} = \frac{3}{4} \times \frac{1}{2} = \frac{3}{8} = 0.375$,
3. $Pr((g(x_r) > g(x)) \wedge (g(x_r) > g(\breve{x}))) = Pr(g(x_r) \geq \max\{g(x), g(\breve{x})\}) = \frac{1}{4} = 0.25$.

Hence, by assuming g is a monotone function, the opposite point has 12.5% $(0.375 - 0.25 = 0.125)$ higher chance to have a higher g value compared to the second random guess. For the following Central Opposition Theorems (one and D-dimensional), no assumption regarding the g function will be made. □

Theorem 4. Central Opposition Theorem for one-dimensional space

Assume

(a) $y = f(x)(x \in [a, b])$ *is an unknown function with at least one solution* $x_s \in [a, b]$ *for* $f(x) = \alpha$; *the solution can be anywhere in our search space (i.e., a black-box optimization problem),*

(b) x *is the first uniform random guess and* x_r *is the second uniform random guess in* $[a, b]$; *candidate solutions should be uniform random numbers because all points have the same chance to be the solution,*

(c) *Opposite of* $x \in [a, b]$ *is defined as* $\breve{x} = a + b - x$,

Then

$Pr(|\breve{x} - x_s| < |x_r - x_s|) > Pr(|x_r - x_s| < |\breve{x} - x_s|)$.

In other words, the probability that the opposite point is closer to the solution is higher than probability of a second random guess.

Proof. In order to prove this theorem, we follow an exhaustive proof by covering all possible situations. Lets say, N different points over the interval $[a, b]$ divide it to $N + 1$ sub-intervals. So, three points $(x \neq (a + b)/2 \neq \breve{x})$ divide the interval $[a, b]$ to four sub-intervals $[a, x]$, $[x, (a + b)/2]$, $[(a + b)/2, \breve{x}]$, and $[\breve{x}, b]$. The solution x_s and the second random guess x_r can form 16 (4×4) different ways/combinations in the above mentioned four sub-intervals. Fig. 3 illustrates all possible situations for a black-box optimization problem. We will call each

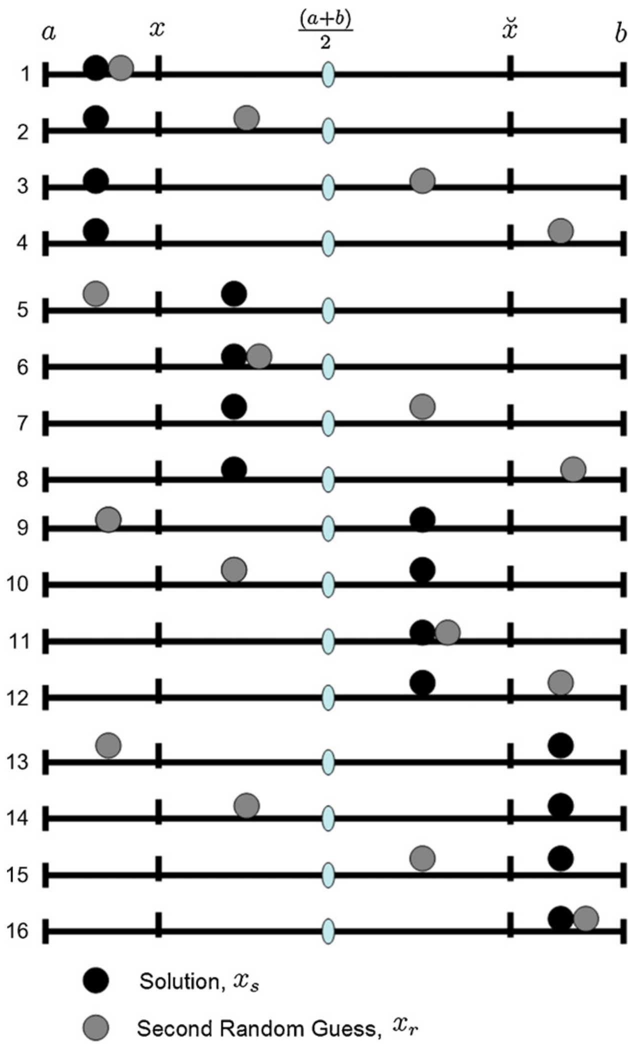

Fig. 3. All possible situations of x_r and x_s for a black-box optimization problem.

situation an event. Therefore, we have 16 possible events (e_i, $i = 1, 2, \ldots, 16$). The probability of all events is equal because the solution (x_s), the first random guess (x), and the second random guess (x_r) can appear anywhere in the interval $[a, b]$ for a black-box optimization problem. Hence, □

$$Pr(e_i) = \frac{1}{16}, \quad i = 1, 2, \ldots, 16. \tag{6}$$

In order to establish an exhaustive proof, we start to calculate the following corresponding probabilities for each event:

p_x= probability of x being the closest to the solution x_s among $\{x, \breve{x}, x_r\}$,
p_r= probability of the second random guess x_r being the closest to the solution x_s among $\{x, \breve{x}, x_r\}$,
$p_{\breve{x}}$= probability of the opposite point $\breve{x}$ being the closest to the solution x_s among $\{x, \breve{x}, x_r\}$.

Obviously we have

$$p_x + p_{\breve{x}} + p_r = 1. \tag{7}$$

Now, all events are categorized into following four groups, see Fig. 4:

1. Group$_1 = \{e_2, e_3, e_4, e_5, e_{12}, e_{13}, e_{14}, e_{15}\}$, $(x, \breve{x} \in [x_s, x_r])$. At least one of x or $\breve{x}$ is located between the x_s and x_r.
2. Group$_2 = \{e_8, e_9\}$, $(\min|x_s - x_r| \geq |x - x_s|) \vee (\min|x_s - x_r| \geq |\breve{x} - x_s|)$. Minimum distance between x_s and x_r is greater than distance between x_s and $x/\breve{x}$.
3. Group$_3 = \{e_7, e_{10}\}$, $(x_s \in [x, (a+b)/2]) \wedge (x_r \in [(a+b)/2, \breve{x}])$ or vice versa.
4. Group$_4 = \{e_1, e_6, e_{11}, e_{16}\}$, x_r and x_s are in the same interval.

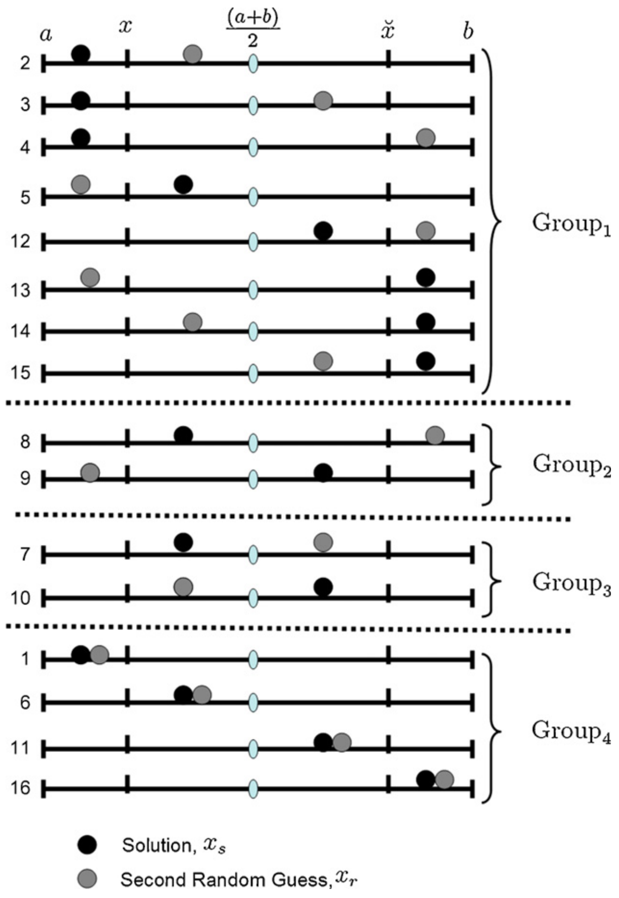

Fig. 4. Similar events are in the same group.

In order to complete our table of probabilities step by step, the corresponding probabilities (p_x, p_r, and $p_{\breve{x}}$) are calculated for each group as follows:

Group$_1$: $\{e_2, e_3, e_4, e_5, e_{12}, e_{13}, e_{14}, e_{15}\}$

When $x \in [x_s, x_r]$ and it is closer to solution than $\breve{x}$, then x is clearly the closest to the solution (events: e_2, e_3, e_4, and e_5, Fig. 4). Hence $p_x = 1$. Similarly, the same logic can be applied to $\breve{x}$ (events: e_{12}, e_{13}, e_{14}, and e_{15}, Fig. 4). For these cases the entries can be inserted into the table of probabilities (Table 2). Newly added values are highlighted in boldface.

Group$_2$: $\{e_8, e_9\}$

(1) If $\min|x_s - x_r| \geq |x - x_s|$, so obviously $p_x = 1$, applicable to e_8.
(2) Similarly, if $\min|x_s - x_r| \geq |\breve{x} - x_s|$, then obviously $p_{\breve{x}} = 1$, applicable to e_9.

These cases are completed in Table 2.

Group$_3$: $\{e_7, e_{10}\}$

Events e_7 and e_{10} are similar cases (Fig. 4). Let us assume $p_x = \alpha$ for event e_7, so we have $p_r = 1 - \alpha$ because $p_x + p_{\breve{x}} + p_r = 1$ and $p_{\breve{x}} = 0$. Analogously, for event e_{10}, we have $p_{\breve{x}} = \alpha$, $p_r = 1 - \alpha$, and $p_x = 0$. We can complete our table for another two events (e_7 and e_{10}), see Table 2.

Now, let us have a preliminary estimate for α. Similar to the reason which was mentioned for Group$_2$, we can conclude

$$\alpha > \frac{1}{2}. \tag{8}$$

If $\min|x_r - x_s| \geq |x - x_s|$, so obviously $p_x = 1$, this case happens with a probability of at least 1/2 when x_s is in the interval $[x, k]$ (the half of the interval $[x, (a+b)/2]$), see Fig. 5.

Table 2
Probabilities table after adding computations of Group$_1$, Group$_2$, and Group$_3$ (left to right, respectively)

Event	p_{x_i}	p_{r_i}	$p_{\breve{x}_i}$	p_{x_i}	p_{r_i}	$p_{\breve{x}_i}$	p_{x_i}	p_{r_i}	$p_{\breve{x}_i}$
e_1	–	–	–	–	–	–	–	–	–
e_2	**1**	**0**	**0**	1	0	0	1	0	0
e_3	**1**	**0**	**0**	1	0	0	1	0	0
e_4	**1**	**0**	**0**	1	0	0	1	0	0
e_5	**1**	**0**	**0**	1	0	0	1	0	0
e_6	–	–	–	–	–	–	–	–	–
e_7	–	–	–	–	–	–	α	**$(1-\alpha)$**	**0**
e_8	–	–	–	**1**	**0**	**0**	1	0	0
e_9	–	–	–	**0**	**0**	**1**	0	0	1
e_{10}	–	–	–	–	–	–	**0**	**$(1-\alpha)$**	α
e_{11}	–	–	–	–	–	–	–	–	–
e_{12}	**0**	**0**	**1**	0	0	1	0	0	1
e_{13}	**0**	**0**	**1**	0	0	1	0	0	1
e_{14}	**0**	**0**	**1**	0	0	1	0	0	1
e_{15}	**0**	**0**	**1**	0	0	1	0	0	1
e_{16}	–	–	–	–	–	–	–	–	–

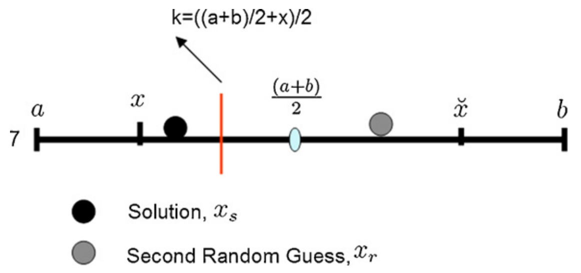

Fig. 5. Illustration of why the inequality $\alpha > 1/2$holds. k is the center of the interval $[x, (a+b)/2]$.

Group$_4$: $\{e_1, e_6, e_{11}, e_{16}\}$

For this group, x_r and x_s are in the same interval. We just need to solve one of them. Lets select event e_1; this case can be decomposed to four possible sub-cases, presented in Fig. 6. For these sub-cases, the probabilities are given in Table 2. (Note the recursive definition of the event (1b)). In order to calculate p_{1x}, let

$$p_{1x} = \frac{1}{4} \times p_{1x} + \frac{1}{4} \times \alpha \Rightarrow p_{1x} = \frac{\alpha}{3}. \tag{9}$$

We know $p(e_i) = 1/16$ (Eq. (6)), so

$$p_{x_1} = \frac{1}{16} \times \frac{\alpha}{3} \Rightarrow p_{1x} = \frac{\alpha}{48}. \tag{10}$$

And finally

$$p_{r_1} = 1 - p_{x1} = \frac{1-\alpha}{48} = \frac{48-\alpha}{48}. \tag{11}$$

Now, we are ready to complete our table (see Table 2). According to our final probabilities table, we have

$$p_x = \sum_{i=1}^{16} p(e_i) p_{x_i} = \frac{5 + 25\alpha/24}{16}. \tag{12}$$

$$p_r = \sum_{i=1}^{16} p(e_i) p_{r_i} = \frac{6 - 50\alpha/24}{16}. \tag{13}$$

$$p_{\breve{x}} = \sum_{i=1}^{16} p(e_i) p_{\breve{x}_i} = \frac{5 + 25\alpha/24}{16}. \tag{14}$$

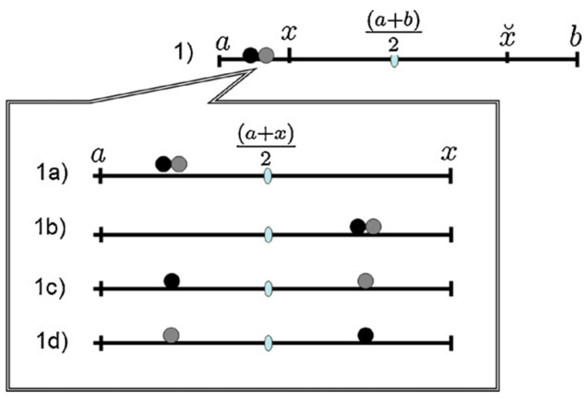

Fig. 6. Four possible sub-cases when x_r and x_s are in the same interval.

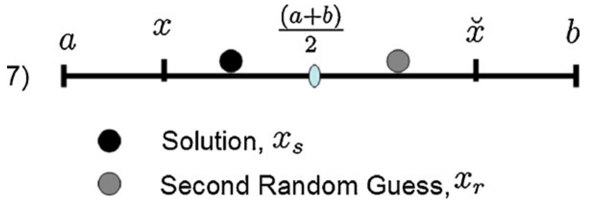

Fig. 7. The e_7 is selected to calculate an impact boundary for the α.

Now, let us investigate when $p_{\breve{x}} > p_r$:

$$(p_{\breve{x}} > p_r) \Leftrightarrow \frac{5 + 25\alpha/24}{16} > \frac{6 - 50\alpha/24}{16} \tag{15}$$

or

$$(p_{\breve{x}} > p_r) \Leftrightarrow \alpha > \frac{24}{75} \tag{16}$$

This is confirmed with $\alpha > 1/2$ (Eq. (8)).

How much better is the opposite point?

Now we want to calculate an impact boundary for α and estimate the value of p_x, $p_{\breve{x}}$, and p_r. In the following two steps, we will find the lower and the upper boundaries for α.

Step 1. Calculating a lower boundary for α—Without lose of generality, we select e_7 to find the impact interval for α (Fig. 7).

As mentioned before, if $\min |x_s - x_r| \geq |x - x_s|$, so obviously $p_x = 1$. As illustrated in Fig. 8, we have

$$p_x|_{\min |x_s - x_r| \geq |x - x_s|} = \lim_{N\to\infty} \sum_{i=1}^{N} P_{s_i} \times P_{r_i}. \tag{17}$$

P_{s_i} and P_{r_i} denote the probability of the presence of the solution and the second random guess in the shown intervals.

$$p_x|_{\min |x_s - x_r| \geq |x - x_s|} = \lim_{N\to\infty} \left(\frac{1}{2^1} \times \frac{1}{2^0} + \frac{1}{2^2} \times \frac{1}{2^1} + \frac{1}{2^3} \times \frac{1}{2^2} + \ldots + \frac{1}{2^{(N+1)}} \times \frac{1}{2^N} \right) \tag{18}$$

$$p_x|_{\min |x_s - x_r| \geq |x - x_s|} = \lim_{N\to\infty} \left(\frac{1}{2^1} + \frac{1}{2^3} + \frac{1}{2^5} + \ldots + \frac{1}{2^{(2N+1)}} \right) \tag{19}$$

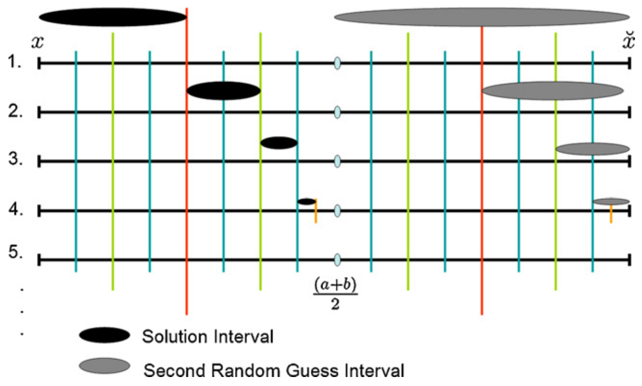

Fig. 8. Situations which $\min |x_s - x_r| \geq |x - x_s|$.

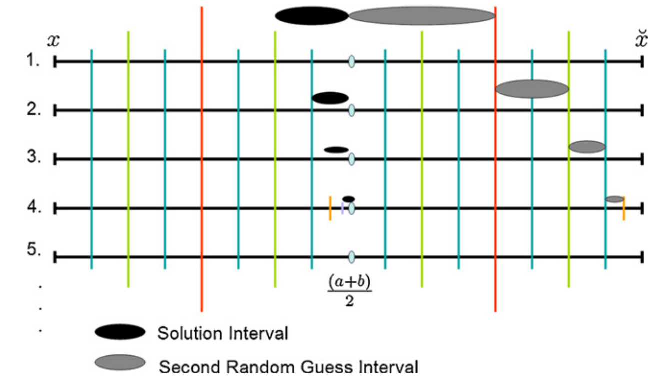

Fig. 9. Situations which $\min|x - x_s| \geq |x_r - x_s|$.

This equation presents infinite geometric series; such series converge if and only if the absolute value of the common ratio is less than one ($|r| < 1$). For these kinds of series we have

$$\lim_{N\to\infty}\sum_{k=1}^{N} ar^k = \lim_{N\to\infty}\frac{a(1-r^{N+1})}{1-r} = \frac{a}{1-r}. \tag{20}$$

Hence

$$p_{x|_{\min|x_s-x_r|\geq|x-x_s|}} = \frac{4}{6}. \tag{21}$$

So, we receive

$$\frac{4}{6} \leq \alpha. \tag{22}$$

Step 2. Calculating an upper boundary for α—Analogously, If $\min|x-x_s| \geq |x_s - x_r|$, then $p_r = 1$. So, as illustrated in Fig. 9, we have

$$p_{r|_{\min|x-x_s|\geq|x_s-x_r|}} = \lim_{N\to\infty}\sum_{i=1}^{N} P_{s_i} \times P_{r_i} \tag{23}$$

$$p_{r|_{\min|x-x_s|\geq|x_s-x_r|}} = \lim_{N\to\infty}\left(\frac{1}{2^2}\times\frac{1}{2^1}+\frac{1}{2^3}\times\frac{1}{2^2}+\frac{1}{2^4}\times\frac{1}{2^3}+\ldots+\frac{1}{2^{(N)}}\times\frac{1}{2^{(N-1)}}\right) \tag{24}$$

$$p_{r|_{\min|x-x_s|\geq|x_s-x_r|}} = \lim_{N\to\infty}\left(\frac{1}{2^3}+\frac{1}{2^5}+\frac{1}{2^7}+\ldots+\frac{1}{2^{(2N-1)}}\right) \tag{25}$$

Again, we are faced with infinite geometric series and by reusing the Eq. (20), we have

$$p_{r|_{\min|x-x_s|\geq|x_s-x_r|}} = \frac{1}{6}. \tag{26}$$

Finally, we have

$$\alpha \leq (1 - p_{r|_{\min|x-x_s|\geq|x_s-x_r|}}) = \frac{5}{6}, \tag{27}$$

$$p_{x|_{\min|x_s-x_r|\geq|x-x_s|}} \leq \alpha \leq (1 - p_{r|_{\min|x-x_s|\geq|x_s-x_r|}}), \tag{28}$$

or

$$\frac{4}{6} \leq \alpha \leq \frac{5}{6}. \tag{29}$$

By establishing this boundaries for α and considering Eqs. (12)–(14), we have

$$\frac{205}{576} \leq p_x = p_{\breve{x}} \leq \frac{845}{2304}. \tag{30}$$

or

$$0.36 \leq p_x = p_{\breve{x}} \leq 0.37. \tag{31}$$

And also

$$\frac{307}{1152} \leq p_r \leq \frac{83}{288}. \tag{32}$$

or

$$0.27 \leq p_r \leq 0.29. \tag{33}$$

Hence, the opposite of x ($\breve{x}$) has a higher chance to be closer to the solution, x_s, compared to the second random guess, x_r, in a one-dimensional solution space.

The center of the interval $[\frac{4}{6}, \frac{5}{6}]$ for α is 9/12 or 0.75. By substituting this mean value in Eqs. (12)–(14), we receive

$$p_x = p_{\breve{x}} = 0.3613 \quad \text{and} \quad p_r = 0.2773. \tag{34}$$

Central Opposition Theorem can be extended to higher dimensions, following theorem addresses this extension.

Theorem 5. Central Opposition Theorem for D-dimensional space *Assume*

(a) *$y = f(X)$is an unknown function with $X(x_1, x_2, x_3, \ldots, x_D)$, $x_i \in [a_i, b_i]$, $i = 1, 2, 3, \ldots, D$ and at least one solution at $X_s(x_{s_1}, x_{s_2}, x_{s_3}, \ldots, x_{s_D})$, $x_{s_i} \in [a_i, b_i]$, $i = 1, 2, 3, \ldots, D$,*
(b) *X is the first uniform random guess and $X_r(x_{r_1}, x_{r_2}, x_{r_3}, \ldots, x_{r_D})$ is the second uniform random guess in the solution space,*
(c) *The opposite point of $X(x_1, x_2, \ldots, x_D)$ is defined by $\breve{X}(\breve{x}_1, \breve{x}_2, \ldots, \breve{x}_D)$ where*

$\breve{x}_i = a_i + b_i - x_i, i = 1, 2, 3, \ldots, D.$

Then $Pr(\|\breve{X}, X_s\| < \|X_r, X_s\|) > Pr(\|X_r, X_s\| < \|\breve{X}, X_s\|)$, where $\|\cdot\|$ denotes the Euclidean distance.

Proof. We have

$$Pr(\|\breve{X}, X_s\| < \|X_r, X_s\|) > Pr(\|X_r, X_s\| < \|\breve{X}, X_s\|)$$
$$= Pr\left(\sqrt{\sum_{i=1}^{D}(\breve{x}_i - x_{s_i})^2} < \sqrt{\sum_{i=1}^{D}(x_{r_i} - x_{s_i})^2}\right)$$
$$> Pr\left(\sqrt{\sum_{i=1}^{D}(x_{r_i} - x_{s_i})^2} < \sqrt{\sum_{i=1}^{D}(\breve{x}_i - x_{s_i})^2}\right)$$

According to the Central Opposition Theorem for one-dimensional space we have

$$Pr(|\breve{x} - x_s| < |x_r - x_s|) > Pr(|x_r - x_s| < |\breve{x} - x_s|). \tag{35}$$

That is true for each dimension in the solution space, so

$$Pr(|\breve{x_i} - x_{s_i}| < |x_{r_i} - x_{s_i}|) > Pr(|x_{r_i} - x_{s_i}| < |\breve{x_i} - x_{s_i}|), \quad i = 1, 2, 3, \ldots, D \tag{36}$$

Hence

$$Pr\left(\sqrt{\sum_{i=1}^{D}(\breve{x_i} - x_{s_i})^2} < \sqrt{\sum_{i=1}^{D}(x_{r_i} - x_{s_i})^2}\right) > Pr\left(\sqrt{\sum_{i=1}^{D}(x_{r_i} - x_{s_i})^2} < \sqrt{\sum_{i=1}^{D}(\breve{x_i} - x_{s_i})^2}\right),$$

and the Central Opposition Theorem is also valid for a *D*-dimensional space. □

Simply, if in each dimension $\breve{x_i}$ has a higher chance to be closer to the solution than x_i, consequently according to Euclidean distance, the point $\breve{P}$, in a *D*-dimensional space, will have a higher chance to be closer to solution than *P*.

Algorithm 1. Calculate p_x, $p_{\breve{x}}$, and p_r experimentally.

```
1: $D_{max} \leftarrow 10,000$
2: $n \leftarrow 1,000,000$
3: for $D = 1$ to $D_{max}$ do
4:   for $R = 1$ to $n$ do
5:     Generate three random points $X$, $X_s$, $X_r$ in the D-dimensional space, $[a_i, b_i] = [-1, 1]$
       for $i = 1, 2, 3, ..., D$
6:     Calculate the opposite point of $X$ ($\breve{X}$)
7:     Calculate the Euclidean distance of $X$, $\breve{X}$, and $X_r$ from $X_s$ ($d_X$, $d_{\breve{X}}$, $d_r$)
8:     if $(d_X < d_{\breve{X}}) \wedge (d_X < d_r)$ then
9:       $c_x \leftarrow c_x + 1$ {$x$ is the closest to the solution}
10:    else if $(d_{\breve{X}} < d_X) \wedge (d_{\breve{X}} < d_r)$ then
11:      $c_{\breve{x}} \leftarrow c_{\breve{x}} + 1$ {$\breve{x}$ is the closest to the solution}
12:    else if $(d_r < d_X) \wedge (d_r < d_{\breve{X}})$ then
13:      $c_r \leftarrow c_r + 1$ {$x_r$ is the closest to the solution}
14:    end if
15:  end for
16:  $p_x \leftarrow c_x/n$
17:  $p_{\breve{x}} \leftarrow c_{\breve{x}}/n$
18:  $p_r \leftarrow c_r/n$
19: end for
```

3. Empirical verification of mathematical results

In this section, the aforementioned mathematical proofs are experimentally verified and the usefulness of the opposite numbers in higher dimensional spaces is investigated. For this propose, three random points in a *D*-dimensional space are generated (*n* times), called *X*, X_s, and X_r. Then, the number of times (out of *n*) which *X*, $\breve{X}$, or X_r is the closest to the randomly generated solution X_s (measured by Euclidean distance) is counted and finally the probability of the closeness of each point is calculated (p_x, $p_{\breve{x}}$, and p_r). In conducted experiments, *n* is chosen a large number in order to have an accurate estimation for probability values. The proposed method for this empirical verification is presented in Algorithm 1. As shown, there are two nested loops, the outer one to feed dimensions and the inner one to handle *n* trials for each dimension.

The experiments have been conducted for different dimensions ranging from $D = 1$ to $D = 10,000$. In order to attain reliable results, the number of trials *n* was set to 1,000,000. Results are summarized in Table 3.

Results analysis: The mean μ (across all dimensions), standard deviation σ, and 95% confidential interval (CI) of p_x, $p_{\breve{x}}$, and p_r have been calculated for the results of dimensions $D = 1, 2, 3, \ldots, 10{,}000$. Low standard deviations and short confidence intervals show that the probabilities remain the same for all investigated dimensions. The probability of opposite point $p_{\breve{x}}$ (0.3617) is 0.085 higher than the probability of a second random guess p_r (0.2767). As shown in Table 3, interestingly, experimental results conform with established theorems.

Additional experiments (not presented here) showed that $\alpha = 0.75$ is a proper value (a similar experimental method presented in this section is used to simulate e_7 and calculate α, see Fig. 7). Using this empirical value in Eqs. (12)–(14), a more accurate comparison between theoretical and experimental probabilities can be provided (see Table 3). As seen, the probabilities are almost the same.

Table 3
Probabilities table of shown four cases in Fig. 6

Event	p_{1x_i}	p_{1r_i}	$p_{1\breve{x}_i}$
1a	0	1	0
1b	p_{1x}	p_{1r}	0
1c	0	1	0
1d	α	$(1-\alpha)$	0

4. Employing OBL to accelerate differential evolution

Differential evolution (DE) was proposed by Price and Storn in 1995 [29,31]. It is an effective, robust, and simple global optimization algorithm [36] which has only a few control parameters. According to frequently reported comprehensive studies [19,32,33,36], DE outperforms many other optimization methods in terms of convergence speed and robustness over common benchmark functions and real-world problems. Generally speaking, all population-based optimization algorithms, no exception for DE, suffer from long computational times because of their evolutionary/stochastic nature. This crucial drawback sometimes limits their application to off-line problems with little or no realtime constraints.

In this section, OBL has been utilized to accelerate the convergence rate of differential evolution. Hence, our proposed approach has been called opposition-based differential evolution (ODE). ODE uses opposite numbers during population initialization and also for generating new populations during the evolutionary process. To the best of our knowledge, this is the first time that opposite numbers have been utilized to speed up the convergence rate of an optimization algorithm.

4.1. Opposition-based differential evolution (ODE)

The flowchart of the classical differential evolution (DE) are presented in Fig. 10 by white blocks. The grey blocks (1) and (3) indicate the opposition-based population initialization and generation jumping, respectively, which are embedded inside the classical DE algorithm to accelerate that using the OBL. These two blocks are explained in the following sections.

4.1.1. Opposition-based population initialization

According to authors best knowledge, random number generation, in absence of a priori knowledge, is the commonly used choice to create an initial population. But, as mentioned before, by utilizing OBO we can obtain fitter starting candidates even when there is no a priori knowledge about the solution(s). Block (1) from Fig. 10 shows the implementation of opposition-based population initialization. Following steps explain that procedure:

Step 1. Initialize the population $P(N_p)$ randomly,

Step 2. Calculate opposite population by

$$\mathrm{OP}_{i,j} = a_j + b_j - P_{i,j}, \quad i = 1, 2, \ldots, N_p; j = 1, 2, \ldots, D. \tag{37}$$

where $P_{i,j}$ and $\mathrm{OP}_{i,j}$ denote the j th variable of the i th population and the opposite-population vector, respectively.

Step 3. Select the N_p fittest individuals from the set $\{P \cup \mathrm{OP}\}$ as the initial population.

According to the above procedure, $2N_p$ function evaluations are required instead of N_p for the regular random population initialization. But, by the opposition-based initialization, the parent algorithm can start with the fitter initial individuals instead, and this is a one-time cost.

4.1.2. Opposition-based generation jumping

By applying a similar approach to the current population, the evolutionary process can be forced to jump to a fitter generation. Based on a jumping rate J_r (i.e., jumping probability), after generating new populations by mutation, crossover, and selection, the opposite population is calculated

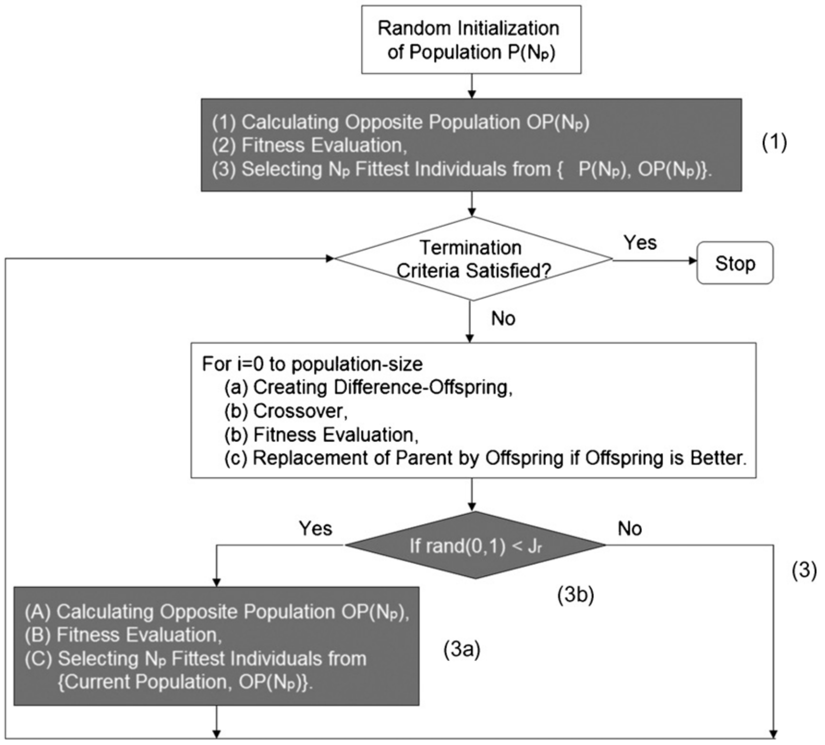
Random Initialization of Population P(Np)
(1) Calculating Opposite Population OP(Np)
(2) Fitness Evaluation,
(3) Selecting Np Fittest Individuals from { P(Np), OP(Np)}.
(1)
Termination Criteria Satisfied?
Yes
Stop
No
For i=0 to population-size
(a) Creating Difference-Offspring,
(b) Crossover,
(b) Fitness Evaluation,
(c) Replacement of Parent by Offspring if Offspring is Better.
Yes
If rand(0,1) < Jr
No
(3)
(3b)
(A) Calculating Opposite Population OP(Np),
(B) Fitness Evaluation,
(C) Selecting Np Fittest Individuals from {Current Population, OP(Np)}.
(3a)

Fig. 10. Opposition-based differential evolution (ODE).

and the N_p fittest individuals are selected from the union of the current population and the opposite population. As a difference to opposition-based initialization, it should be noted here that in order to calculate the opposite population for generation jumping, the opposite of each variable is calculated dynamically. That is, the maximum and minimum values of each variable in the *current population* ($[\mathrm{MIN}_j^p, \mathrm{MAX}_j^p]$) are used to calculate opposite points instead of using variables' predefined interval boundaries ($[a_j, b_j]$):

$$\mathrm{OP}_{i,j} = \mathrm{MIN}_j^p + \mathrm{MAX}_j^p - P_{i,j}, \quad i = 1, 2, \ldots, N_p; \quad j = 1, 2, \ldots, D. \tag{38}$$

By staying within variables' static boundaries, it is possible to jump outside of the already shrunken search space and lose the knowledge of the current reduced space (converged population). Hence, we calculate opposite points by using variables' current interval in the population ($[\mathrm{MIN}_j^p, \mathrm{MAX}_j^p]$) which is, as the search does progress, increasingly smaller than the corresponding initial range $[a_j, b_j]$. Block (3) from Fig. 10 indicates the implementation of opposition-based generation jumping.

4.2. Benchmark functions

A set of 15 benchmark functions (7 unimodal and 8 multimodal functions) has been used for performance verification of the proposed approach. Furthermore, test functions with two different dimensions (D and $2 \times D$) have been employed in the conducted experiments. By this way, the classical differential evolution (DE) and opposition-based DE (ODE) are compared on 30 minimization problems. The definition of the benchmark functions is given in Table 4.2.

4.3. Comparison strategies and metrics

In this study, three metrics, namely, number of function calls (NFC), success rate (SR), and success performance (SP) [35] have been utilized to compare the algorithms. We compare the convergence speed by measuring the number of function calls which is the most commonly used metric in literature [11–14,35]. A smaller NFC means higher convergence speed. The termination criterion is to find a value smaller than the value-to-reach (VTR) before reaching the maximum number of function calls $\mathrm{MAX}_{\mathrm{NFC}}$. In order to minimize the effect of the stochastic nature of the algorithms on the metric, the reported number of function calls for each function is the average over 50 trials.

Table 4
Probabilities table, final result

Event	p_{x_i}	p_{r_i}	$p_{\breve{x}_i}$
e_1	$(\alpha/48)$	$(48-\alpha)/48$	0
e_2	1	0	0
e_3	1	0	0
e_4	1	0	0
e_5	1	0	0
e_6	$(\alpha/48)$	$(48-\alpha)/48$	0
e_7	α	$(1-\alpha)$	0
e_8	1	0	0
e_9	0	0	1
e_{10}	0	$(1-\alpha)$	α
e_{11}	0	$(48-\alpha)/48$	$(\alpha/48)$
e_{12}	0	0	1
e_{13}	0	0	1
e_{14}	0	0	1
e_{15}	0	0	1
e_{16}	0	$(48-\alpha)/48$	$(\alpha/48)$
Sum	$(5+25\alpha/24)$	$(6-50\alpha/24)$	$(5+25\alpha/24)$

Table 5
Numerical results generated by Algorithm 1 (μ: Mean, σ: Standard deviation, CI: Confidence interval)

	p_x	$p_{\breve{x}}$	p_r
μ	0.3617	0.3617	0.2767
σ	0.0048	0.0048	0.0045
95%CI	(0.3616, 0.3618)	(0.3616, 0.3618)	(0.2766, 0.2767)

The number of times, for which the algorithm succeeds to reach the VTR for each test function is measured as the success rate SR:

$$\mathrm{SR} = \frac{\text{number of times reached VTR}}{\text{total number of trials}}. \tag{39}$$

The average success rate ($\mathrm{SR}_{\mathrm{ave}}$) over n test functions are calculated as follows:

$$\mathrm{SR}_{\mathrm{ave}} = \frac{1}{n}\sum_{i=1}^{n}\mathrm{SR}_i. \tag{40}$$

Both of NFC and SR are important measures in an optimization process. So, two individual objectives should be considered simultaneously to compare competitors. In order to combine these two metrics, a new measure, called success performance (SP), has been introduced as follows [35]:

$$\mathrm{SP} = \frac{\text{mean (NFC for successful runs)}}{\mathrm{SR}}. \tag{41}$$

SP is our main measure to judge which algorithm performs better than others.

4.4. Setting control parameters

Parameter settings for all conducted experiments are presented in Table 4.4. The same setting has been used in literature cited after of each parameter.

Table 6
Comparison of experimental and mathematical results

	p_x	$p_{\breve{x}}$	p_r
Mathematical computation	(0.3559, 0.3667)	(0.3559, 0.3667)	(0.2664, 0.2881)
Experimental results	0.3617	0.3617	0.2767

Table 7
Comparison of experimental and mathematical results for $\alpha = 0.75$

	p_x	$p_{\breve{x}}$	p_r
Mathematical computation ($\alpha = 0.75$)	0.3613	0.3613	0.2773
Experimental results	0.3617	0.3617	0.2767

Table 8
List of employed benchmark functions

Name	Definition	S
1st De Jong	$f_1(X) = \sum_{i=1}^{D} x_i{}^2$	$[-2.56, 7.68]^D$
Axis parallel hyper-ellipsoid	$f_2(X) = \sum_{i=1}^{D} i x_i{}^2$	$[-2.56, 7.68]^D$
Schwefel's Problem 1.2	$f_3(X) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j\right)^2$	$[-32.5, 97.5]^D$
Rastrigin's function	$f_4(X) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i)\right)$	$[-2.56, 7.68]^D$
Griewangk's function	$f_5(X) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-300, 900]^D$
Sum of different power	$f_6(X) = \sum_{i=1}^{D} \lvert x_i\rvert^{(i+1)}$	$[-0.5, 1.5]^D$
Ackley's problem	$f_7(X) = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i^2}{D}}\right) - \exp\left(\frac{\sum_{i=1}^{D}\cos(2\pi x_i)}{D}\right) + 20 + e$	$[-16, 48]^D$
Levy function	$f_8(X) = \sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i - 1)^2(1 + \sin^2(3\pi x_{i+1})) + (x_D - 1)(1 + \sin^2(2\pi x_D))$	$[-10, 10]^D$
Michalewicz function	$f_9(X) = -\sum_{i=1}^{D} \sin(x_i)(\sin(i x_i^2/\pi))^{2m}, \quad (m = 10)$	$[0, \pi]^D$
Zakharov function	$f_{10}(X) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^4$	$[-5, 10]^D$
Schwefel's Problem 2.22	$f_{11}(X) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	$[-5, 15]^D$
Step function	$f_{12}(X) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-50, 150]^D$
Alpine function	$f_{13}(X) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1 x_i\rvert$	$[-5, 15]^D$
Exponential problem	$f_{14}(X) = \exp\left(-0.5\sum_{i=1}^{D} x_i^2\right)$	$[-0.5, 1.5]^D$
Salomon problem	$f_{15}(X) = 1 - \cos(2\pi\lVert x\rVert) + 0.1\lVert x\rVert$, where $\lVert x\rVert = \sqrt{\sum_{i=1}^{D} x_i^2}$	$[-50, 150]^D$

The 13 functions (out of 15) have an optimum in the center of searching space, to make it asymmetric, the search space for all of these functions are shifted $a/2$ (which means if $-a \le x_i \le a$ and $f_{\min} = f(0, \ldots, 0) = 0$ then $-a + a/2 \le x_i \le a + a/2$).

Table 9
Parameter settings

Parameter name	Setting	Reference
Population size (N_p)	100	[13,37–39]
Differential amplification factor (F)	0.5	[13,29,30,32,38,40]
Crossover probability constant (C_r)	0.9	[13,29,30,32,38,40]
Jumping rate constant (J_r)	0.3	[11–14]
Maximum number of function calls (MAX_{NFC})	10^6	[11,13,14]
Value to reach (VTR)	10^{-8}	[13,35]
Mutation strategy	DE/rand/1/bin	[22,29,36,38,41]

4.5. Experimental results

Results of applying DE and ODE to solve 30 test problems are given in Table 4.5. The best function call (NFC) and the success performance for each case are highlighted in boldface. As seen, ODE outperforms DE on 25 functions, while, DE just on 4 functions shows better results than ODE. DE performs marginally better than ODE in terms of average success rate (0.90 versus 0.88). As we mentioned before, the success performance is a measure which considers the number of function calls and the success rate simultaneously and so it can

Table 10
Comparison of DE and ODE

F	D	DE			ODE		
		NFC	SR	SP	NFC	SR	SP
f_1	30	86,072	1	86,072	**50,844**	1	**50,844**
	60	15,4864	1	15,4864	**101,832**	1	**101,832**
f_2	30	95,080	1	95,080	**56,944**	1	**56,944**
	60	17,6344	1	17,6344	**117,756**	1	**117,756**
f_3	20	**174,580**	1	**174,580**	177,300	1	177,300
	40	**816,092**	1	**816,092**	834,668	1	834,668
f_4	10	323,770	0.96	337,260	**75,278**	0.92	**81,823**
	20	811,370	0.08	10,142,125	**421,300**	0.16	**2,633,125**
f_5	30	111,440	0.96	116,083	**74,717**	0.92	**81,214**
	60	193,960	1	193,960	**128,340**	0.68	**188,735**
f_6	30	18,760	1	18,760	**10,152**	1	**10,152**
	60	33,128	1	33,128	**11,452**	1	**11,452**
f_7	30	168,372	1	168,372	**100,280**	1	**100,280**
	60	294,500	1	294,500	**202,010**	0.96	**210,427**
f_8	30	101,460	1	101,460	**70,408**	1	**70,408**
	60	180,260	0.84	215,000	**121,750**	0.60	**202,900**
f_9	10	**191,340**	0.76	**252,000**	213,330	0.56	380,900
	20	288,300	0.35	824,000	**253,910**	0.55	**461,700**

Table 10 (*Continued*)

F	*D*	DE			ODE		
		NFC	SR	SP	NFC	SR	SP
f_{10}	30	385,192	1	385,192	**369,104**	1	**369,104**
	60	–	0	–	–	0	–
f_{11}	30	183,408	1	183,408	**167,580**	1	**167,580**
	60	318,112	1	318,112	**274,716**	1	**274,716**
f_{12}	30	40,240	1	40,240	**26,400**	1	**26,400**
	60	73,616	1	73,616	**64,780**	1	**64,780**
f_{13}	30	386,920	1	386,920	**361,884**	1	**361,884**
	60	432,516	1	**432,516**	**425,700**	0.96	443,438
f_{14}	10	19,324	1	19,324	**16,112**	1	**16,112**
	20	45,788	1	45,788	**31,720**	1	**31,720**
f_{15}	10	37,260	1	37,260	**26,108**	1	**26,108**
	20	17,6872	1	17,6872	**57,888**	1	**57,888**
	SR_{ave}		0.90			0.88	

D: dimension, NFC: number of function calls (average over 50 trials), SR: success rate, SP: success performance. The last row of the table presents the average success rates. The best NFC and the success performance for each case are highlighted in boldface.

be utilized for a reasonable comparison of optimization algorithms.

5. Conclusion

For many soft computing techniques, in absence of a-priori information about the solution, pure random guess is usually the only option to generate candidate solutions. Obviously the computation time, among others, is directly related to the distance of the guess from the optimal solution.

Experimental results, recently published, indicate that employing opposition-based learning within existing soft computing algorithms can accelerate the learn and search process. Promising results have been reported for differential evolution, reinforcement learning and feedforward back-propagation netwroks.

In this paper we established mathematical proofs and experimental evidence to verify the advantage of opposite points compared to additional random points when dealing with high-dimensional problems. Both experimental and mathematical results conform with each other; opposite points are more beneficial than additional independent random points. We can conclude that the opposition-based learning can be utilized to accelerate learning and optimization methods since considering the pair x and $\breve{x}$ has apparently a higher fitness probability than pure randomness.

Finding more accurate global optimum(s) in a shorter period of time for complex problems is the main goal of all evolutionary algorithms. Although the opposition concept has a very old history in other sciences, that is the first time that this concept is employed to enhance population-based algorithms . The conducted experiments confirm that ODE presents higher convergence rate than DE.

Furthermore, the proposed opposition-based schemes (i.e., population initialization and generation jumping) work at the population level and leave the evolutionary part of the algorithms untouched. This generality gives higher flexibility to these schemes to be embedded inside other population-based algorithms (such as GAs) for further investigation.

References

[1] I.L. Lopez Cruz, L.G. Van Willigenburg, G. Van Straten, Efficient differential evolution algorithms for multimodal optimal control problems, Appl. Soft Comput. 3 (2003) 97–122.

[2] C.G. Moles a, J.R. Bangaa, K. Keller, Solving nonconvex climate control problems: pitfalls and algorithm performances, Appl. Soft Comput. 5 (2004) 35–44.

[3] H.R. Tizhoosh, Opposition-based learning: a new scheme for machine intelligence, in: International Conference on Computational Intelligence for Modelling Control and Automation (CIMCA-2005), vol. I, Vienna, Austria, (2005), pp. 695–701.

[4] H.R. Tizhoosh, Reinforcement learning based on actions and opposite actions, in: International Conference on Artificial Intelligence and Machine Learning (AIML-2005), Cairo, Egypt, 2005.

[5] H.R. Tizhoosh, Opposition-Based Reinforcement Learning, J. Adv. Comput. Intell. Intell. Inf. 10 (3) (2006) 578–585.

[6] M. Shokri, H.R. Tizhoosh, M. Kamel, Opposition-based Q (λ) algorithm, in: 2006 IEEE World Congress on Computational Intelligence (IJCNN-2006), Vancouver, BC, Canada, (2006), pp. 646–653.

[7] F. Khalvati, H.R. Tizhoosh, M.D. Aagaard, Opposition-based window memorization for morphological algorithms, in: IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 425–430.

[8] H.R. Tizhoosh, M. Shokri, M.S. Kamel, Opposition-based Q (lambda) with non-Markovian update, in: IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 288–295.

[9] M. Ventresca, H, Tizhoosh, Simulated annealing with opposite neighbors, in: IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 186–192.

[10] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, A novel population initialization method for accelerating evolutionary algorithms, J. Comput. Math. Appl. (Elsevier) 53 (10) (2007) 1605–1614.

[11] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution algorithms, in: 2006 IEEE World Congress on Computational Intelligence (CEC-2006), Vancouver, BC, Canada, (2006), pp. 7363–7370.

[12] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution for optimization of noisy problems, in: 2006 IEEE World Congress on Computational Intelligence (CEC-2006), Vancouver, BC, Canada, (2006), pp. 6756–6763.

[13] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution (ODE), J. IEEE Trans. Evol. Comput. (December), in press.

[14] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, Opposition-based differential evolution (ODE) with variable jumping rate, in: IEEE

Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 81–88.

[15] M. Ventresca, H.R. Tizhoosh, Improving the convergence of backpropagation by opposite transfer functions, in: 2006 IEEE World Congress on Computational Intelligence (IJCNN-2006), Vancouver, BC, Canada, (2006), pp. 9527–9534.

[16] H.R. Tizhoosh, A.R. Malisia, Applying opposition-based ideas to the ant colony system, in: IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 182–189.

[17] M. Ventresca, H. Tizhoosh, Opposite transfer functions and backpropagation through time, in: IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 570–577.

[18] H.R. Tizhoosh, M. Mahootchi, K. Ponnambalam, Opposition-based reinforcement learning in the management of water resources, in: IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 217–224.

[19] J. Andre, P. Siarry, T. Dognon, An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization, Adv. Eng. Software 32 (2001) 49–60.

[20] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Longman Publishing Co, USA, 2005,ISBN:0-201-15767-5.

[21] M. Dorigo, V. Maniezzo, A. Colorni, The Ant System: An Autocatalytic Optimizing Process, Technical Report No. 91-016, Politecnico di Milano, Italy, 1991.

[22] G.C. Onwubolu, B.V. Babu, New Optimization Techniques in Engineering, Springer, Berlin, New York, 2004.

[23] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.

[24] T. Ye, S. Kalyanaraman, A recursive random search algorithm for black-box optimization, ACM Sigmetrics Perform. Eval. Rev. 32 (3) (2004) 44–53.

[25] V. Feoktistov, Differential Evolution: In Search of Solutions, Springer, USA, 2006.

[26] T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms, Oxford University Press Inc., USA, 1996.

[27] T. Bäck, U. Hammel, H.-P. Schwefel, Evolutionary computation: comments on the history and current state, J. IEEE Trans. Evol. Comput. 1 (1) (1997) 3–17.

[28] H.-P. Schwefel, Computational Intelligence: Theory and Practice, Springer-Verlag, New York, USA, 2003.

[29] R. Storn, K. Price, Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optimization 11 (1997) 341–359.

[30] J. Liu, J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Comput. – Fusion Foundations, Methodol. Appl. 9 (6) (2005) 1703–1725.

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

[32] J. Vesterstroem, R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, in: Proceedings of the Congress on Evolutionary Computation (CEC-2004), IEEE Publications, vol. 2, San Diego, California, USA, (July 2004), pp. 1980–1987.

[33] O. Hrstka, A. Kučerová, Improvement of real coded genetic algorithm based on differential operators preventing premature convergence, Adv. Eng. Software 35 (2004) 237–246.

[34] F. Sahba, H.R. Tizhoosh, M.M.A. Salama, Application of opposition-based reinforcement learning in image segmentation, IEEE Symposium on Foundations of Computational Intelligence, Honolulu, Hawaii, USA, April 1–5, 2007, pp. 246–251.

[35] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.-p. Chen, A. Auger, S. Tiwari. Problem definitions and evaluation criteria for the CEC-2005: Special session on real-parameter optimization. KanGAL Report #2005005, IIT Kanpur, India, 2005.

[36] K. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series), 1st ed., Springer, 2005 ISBN:3540209506.

[37] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, J. IEEE Trans. Evol. Comput. 3 (2) (1999) 82.

[38] J. Brest, S. Greiner, B. Bošković, M. Mernik, V. Žumer, Evolutionary programming made faster, J. IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.

[39] C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the Lvy probability distribution, J. IEEE Trans. Evol. Comput. 8 (1) (2004) 1–13.

[40] M.M. Ali, A. Törn, Population set-based global optimization algorithms: Some modifications and numerical studies, J. Comput. Operat. Res. 31 (10) (2004) 1703–1725.

[41] J. Sun, Q. Zhang, E.P.K. Tsang, DE/EDA: a new evolutionary algorithm for global optimization, J. Inf. Sci. 169 (2005) 249–262.