2006 IEEE Congress on Evolutionary Computation
Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada
July 16-21, 2006

Opposition-Based Differential Evolution for Optimization of Noisy Problems

Shahryar Rahnamayan, Hamid R. Tizhoosh, Magdy M.A. Salama, *Fellow, IEEE*

***Abstract*— Differential Evolution (DE) is a simple, reliable, and efficient optimization algorithm. However, it suffers from a weakness, losing the efficiency over optimization of noisy problems. In many real-world optimization problems we are faced with noisy environments. This paper presents a new algorithm to improve the efficiency of DE to cope with noisy optimization problems. It employs opposition-based learning for population initialization, generation jumping, and also improving population's best member. A set of commonly used benchmark functions is employed for experimental verification. The details of proposed algorithm and also conducted experiments are given. The new algorithm outperforms DE in terms of convergence speed.**

I. Introduction

Differential Evolution (DE) is a simple, reliable, and efficient optimization algorithm. However, it suffers from dramatic degradation of the efficiency over optimization of noisy problems. Dealing with noisy fitness functions in evolutionary algorithms has been addressed by some authors in this field, such as evolutionary programming (EP) [1], genetic algorithm (GA) [2], particle swarm optimization (PSO) [3], and differential evolution (DE) [4]. *Re-sampling* and *thresholding* are well-known methods to overcome the noisy fitness evaluation [5], [6]. Re-sampling suggests evaluating of the same candidate solution for N times and approximating of the true fitness value by averaging. N should be determined properly to achieve a reasonable tradeoff between accurate evaluation of fitness value and computation cost. Thresholding method is applied on selection step. According to this method, a parent can only be replaced by an offspring if fitness value of offspring is larger than a threshold value τ. The main problem with this method is finding an optimal static value or modified adaptation rule for τ.

This paper presents a new *opposition-based* differential evolution (ODE) algorithm. It improves the efficiency of the conventional DE over optimization of noisy problems by applying opposition-based learning [7]. The main idea behind the opposition-based learning is considering the estimate and opposite estimate (guess and opposite guess) at the same time in order to achieve a better approximation for current candidate solution. It can be useful for noisy environments when the optimal solution is displaced by the noise. In fact, opposite estimate (looking at the opposite side) gives us a second chance to sense the displacement of optimal solutions. By this way, we obtain a dynamic behavior to follow optimal solution in the noisy environment. The idea is applicable to a wide range of optimization methods. Although the proposed idea is embedded in the DE, but is general enough to be applied to many other population-based algorithms. As another outstanding feature, the ODE can be used to optimize noise-free and also noisy functions without any changes. Many other reported algorithms are designed specifically for optimization of noisy problems, and classical DE can outperform them when they are been applying to noise-free functions. For instance, Das et al. [4] showed that Differential Evolution with Random Scale Factor (DE-RSF-TS) and DE-RSF with Stochastic Selection (DE-RSF-SS) perform worse than DE for noise-free functions.

Organization of this paper is as follows: In section II, the concept of opposition-based learning is explained. The classical DE is briefly reviewed in section III. The proposed algorithm is presented in section IV. Experimental results are given in section V. Finally, the work is concluded in section VI.

II. Opposition-Based Learning

Generally speaking, evolutionary optimization methods start with some initial solutions (initial population) and try to improve performance toward some optimal solutions. The process of searching terminates when predefined criteria are satisfied. In absence of a priori information about the solution, we start with a *random guess*. Obviously, the computation time is directly related to distance of the guess from optimal solution. We can improve our chance to start with a closer (fitter) solution by checking the opposite solution simultaneously. By doing this, the closer one to solution (say guess or opposite guess) can be chosen as initial solution. In fact, according to probability theory, in 50% of cases the guess is farther to solution than opposite guess; for these cases starting with opposite guess can accelerate convergence. The same approach can be applied not only to initial solutions but also continuously to each solution in the current population. The concept of opposition-based learning was introduced by Tizhoosh [7] and its applications were introduced in [7]–[9]. Before concentrating on opposition-based learning, we need to define opposite numbers [7]:

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

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

Pattern Analysis and Machine Intelligence (PAMI) Research Group, Faculty of Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, N2L 3G1, Canada (phone: 1-(519)-888-4567 ext. 6751, fax: 1-(519)-746-4791, emails: shahryar@pami.uwaterloo.ca; tizhoosh@uwaterloo.ca; msalama@hivolt1.uwaterloo.ca).

0-7803-9487-9/06/$20.00/©2006 IEEE

Similarly, this definition can be extended to higher dimensions as follows [7]:

Definition - Let $P(x_1, x_2, ..., x_n)$ be a point in n-dimensional space, where $x_1, x_2, ..., x_n \in R$ and $x_i \in [a_i, b_i]$ $\forall i \in \{1, 2, ..., n\}$. The opposite point of P is defined by $\breve{P}(\breve{x}_1, \breve{x}_2, ..., \breve{x}_n)$ where:

$$\breve{x}_i = a_i + b_i - x_i. \tag{2}$$

Now, by employing opposite point definition, the opposition-based optimization can be defined as follows:

Opposition-Based Optimization - Let $P(x_1, x_2, ..., x_n)$, a point in an n-dimensional space with $x_i \in [a_i, b_i]$ $\forall i \in \{1, 2, ..., n\}$, be a candidate solution. Assume $f(x)$ is a fitness function which is used to measure candidate optimality. According to opposite point definition, $\breve{P}(\breve{x}_1, \breve{x}_2, ..., \breve{x}_n)$ is the opposite of $P(x_1, x_2, ..., x_n)$. 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.

Before introducing the new opposition-based DE algorithm, the classical DE is briefly reviewed in the following section.

III. The Classical DE

Differential Evolution (DE) is a population-based, efficient, robust, and direct search method [10]. Like other evolutionary algorithms, it starts with an initial population vector, which is randomly generated. Let assume that $X_{i,G}, (i = 1, 2, ..., n)$ are n N_v-dimensional parameter vectors of generation G (n is a constant number which presents the population size) [11]. In order to generate a new population of vectors, for each target vector in population three vectors are randomly selected, and weighted difference of two of them is added to the third one.

For classical DE, the procedure is as follows [11]:

(a) Creating difference-offspring: For each vector i from generation G a mutant vector $V_{i,G}$ is defined by

$$V_{i,G} = X_{r_1,G} + F(X_{r_2,G} - X_{r_3,G}), \tag{3}$$

where $i = \{1, 2, ..., n\}$ and r_1, r_2, and r_3 are mutually different random integer indices selected from $\{1, 2, ..., n\}$. Further, i, r_1, r_2, and r_3 are different so $n \geq 4$. $F \in [0, 2]$ is a real constant which determines amplification of the added differential variation of $(X_{r2,G} - X_{r3,G})$. Larger values for F result in higher diversity in the generated population and the lower values in faster convergence.

DE utilizes crossover operation to increase diversity of the population. It defines following trial vector:

$$U_{i,G} = (U_{1i,G}, U_{2i,G}, ..., U_{N_v i,G}), \tag{4}$$

where $j = 1, 2, .., N_v$ and

$$U_{ji,G} = \begin{cases} V_{ji,G} & \text{if } rand_j(0,1) \leq Cr, \\ X_{ji,G} & \text{otherwise.} \end{cases} \tag{5}$$

$Cr \in (0, 1)$ is predefined crossover constant $rand_j(0, 1) \in [0, 1]$ is jth evaluation of uniform random generator. Most popular value for Cr is in the range of $(0.4, 1)$ [4].

(b) Fitness evaluating of trial vector.

(c) Selection: The approach must decide which vector ($U_{i,G}$ or $X_{i,G}$) should be a member of new generation, $G + 1$. Vector with the fitter value is chosen.

There are other variants of DE [12] but to maintain a general comparison, the classical version of DE has been selected to compare with the proposed algorithm in all conducted experiments.

IV. Proposed Algorithm

In this section, the concept of opposition-based optimization is embedded in the DE to accelerate convergence speed and also to enhance its capability for handling noisy optimization problems. After population initialization, DE algorithm remains inside a loop and continues to produce new generations and stops if termination criterion is satisfied. Population initialization, producing the new generations, and improving the best individual in the current population are three target stages which are extended by opposition-based concept. Flowchart of opposition-base differential evolution (ODE) algorithm is given in Fig. 1 and the corresponding algorithm is presented using a pseudo-code in Table I.

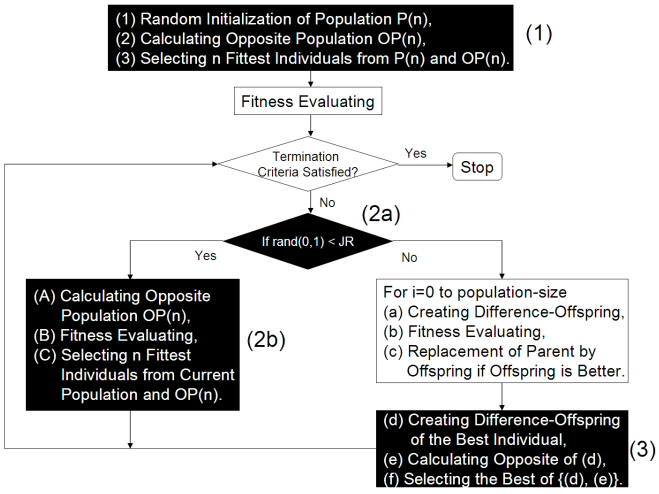


Fig. 1. Opposition-based DE (ODE). New/extended blocks are illustrated by black boxes.

Newly added or changed blocks in the DE are emphasized by black boxes and are explained in details as follows:

TABLE I

OPPOSITION-BASED DE (ODE). NEW/EXTENDED BLOCKS ARE HIGHLIGHTED IN BOLDFACE.

Begin
  n = population size;
  $k = \{1, 2, ..., n\}$; /* index of individuals in the population */
  $j = \{1, 2, ..., N_v\}$; /* index of variables in the individual */
  $x_j \in [a_j, b_j]$; /* interval boundaries of variable j */
  $x^p{}_j \in [a^p{}_j, b^p{}_j]$; /* interval boundaries of variable j in the current population */
  MAX_{NFC} = maximum number of function calls (NFC);
  VTR = value to reach;
  Jr = jumping rate;

  /* (1) Opposition-Based Population Initialization */
  for $k = 0$ to n
    Generating Uniformly Distributed Random Individual, $P_{k,j}$;
    Calculating Corresponding Opposite Individual by $OP_{k,j} = a_j + b_j - P_{k,j}$;
  for end
  Calculating Fitness Value of each Individual in P(n) and OP(n);
  Selecting n Fittest Individuals from $\{P(n), OP(n)\}$ as Initial Population;
  /* End of (1) Opposition-Based Population Initialization */

  while ($Best\_Fitness\_Value\_so\_far >$ VTR **and** NFC $< \text{MAX}_{\text{NFC}}$)
    if (rand (0,1) $<$ Jr) **/* (2a) */**
      /* (2b) Opposition-Based Generation Jumping */
        for $k = 0$ to n
          Calculating Opposite Individual in Current_Population by $OP_{k,j} = a^p{}_j + b^p{}_j - P_{k,j}$;
        for end
      Calculating Fitness Value of each Individual in $OP(n)$;
      Selecting n Fittest Individuals from $\{OP(n), \text{Current\_Population}\}$ as a new Current_Population;
      /* End of (2b) Opposition-Based Generation Jumping */
    else
      /* DE Evolution Steps */
      for $k = 0$ to n
        Creating Difference-Offspring of each Parent (Individual);
        Calculating Fitness Value of above Offspring;
        Selecting the Best from {Parent,Offspring};
      for end
      /* End of DE Evolution Steps */
    if end
      /* (3) Best Individual Jumping */
    Calculating Difference-Offspring of the Best Individual (best), newbest;
    Calculating Opposite of new_best by op_newbest $= a^p{}_j + b^p{}_j -$ newbest;
    Replacing the current population's best member by the fittest member of the set {best,newbest,op_newbest};
      /* End of (3) Best Individual Jumping */
  while end
End

(1) Opposition-Based Population Initialization

According to our review of optimization literature, in most cases, random number generation is the only choice to create initial population. But as mentioned in section II, concept of opposition-based optimization can help us to obtain fitter starting candidate solutions even when there is no a priori knowledge about solutions. We propose the following scheme:

(1) Generating uniformly distributed random population, $P(n)$; n is the population size,

(2) Calculating opposite population $OP(n)$; the k^{th} corresponding opposite individual for $OP(n)$ is calculated by:

$$OP_{k,j} = a_j + b_j - P_{k,j}, \tag{6}$$

$$k = 1, 2, ..., n; j = 1, 2, ..., N_v,$$

where N_v is the number of variables (problem dimension); a_j and b_j denote the interval boundaries of j^{th} variable ($x_j \in [a_j, b_j]$),

(3) Selecting n fittest individuals from set the $\{P(n), OP(n)\}$ as initial population.

(see block (1) in Fig. 1 and also Opposition-Based Population Initialization boldface block in Table I)

(2) Opposition-Based Generation Jumping

Based on a jumping rate Jr, instead of generating new population by evolutionary process, the opposite population is calculated and the n fittest individuals are selected from the current population and the corresponding opposite population (exactly similar to what was performed for opposition-based population initialization). Blocks (2a) and (2b) in Fig. 1 and also Opposition-Based Generation Jumping boldface block in Table I present more details. Our comprehensive experiments show that Jr should be a small number (Jr$\in (0, 0.4)$).

Dynamic Opposition: It should be noted here that in order

to calculate the opposite individuals for generation jumping and also for the best individual jumping (step (3)), the opposite of each variable is calculated dynamically. It means, the maximum and minimum values of each variable in *current population* ($[a^p{}_j, b^p{}_j]$) are used to calculate opposite point instead of using variables' predefined interval boundaries ($[a_j, b_j]$):

$$OP_{k,j} = a^p{}_j + b^p{}_j - P_{k,j}, \quad (7)$$
$$k = 1, 2, ..., n; j = 1, 2, ..., N_v.$$

This dynamic behavior of the opposite point calculation increases our chance to find fitter opposite points. By keeping variables' interval static boundaries, we will jump outside of solution space and the knowledge of current reduced space (converged population) is not utilized to find better opposite candidate.

(3) Best Individual Jumping

In this stage, we locally improve the best candidate (the fittest member) in the current population by applying following steps:

(3.d) Creating difference-offspring of the best individual in the current population by:

$$\text{newbest} = \text{best} + F'(X_{r_1} - X_{r_2}), \quad (8)$$

where r_1 and r_2 are mutually different random integer indices selected from $\{1, 2, ..., n\}$. F' is a real constant which determines amplification of the added differential variation of $(X_{r_1} - X_{r_2})$. F' should be set to a small number ($F' \in (0, 0.2]$) because we need a small/local exploration to improve the current best member. In contrast, a large value for F' can reduced our chance to obtain a better candidate.

(3.e) Calculating opposite of offspring created in (3d) by employing Eq. 7, call op_newbest,

(3.f) Replacing the current best member by the fittest member of the set {best, newbest, op_newbest}.

(see block (3) in Fig. 1 and also Best Individual Jumping boldface block in Table I)

By embedding opposition-based optimization concept in the mentioned three blocks, the convergence speed and the capability of handling noisy optimization problems will be remarkably increased. This is demonstrated in the following section.

V. Experiments

A. Control Parameter Settings

The parameter settings are listed as follows. All common parameters of the DE and ODE are set as the same to have a fair competition. For both DE and ODE we set

- Population size, $n = 100$
- Differential amplification factor, F=0.5
- Crossover probability constant, Cr=0.9
- Strategy [12], $DE/rand/1/bin$
- Maximum function calls, $\text{MAX}_{\text{NFC}}=10^5$.

For ODE we set

- Jumping rate constant, Jr=0.3
- Differential amplification factor, F′=0.1.

B. Benchmark Functions

Following functions are well-known benchmark functions for minimization [4], [6], [13]. The noisy version of each benchmark function, is defined as:

$$f_n(\vec{x}) = f(\vec{x}) + N(0, \sigma^2),$$

where $f(\vec{x})$ is the noise-free function; $f_n(\vec{x})$ is the corresponding noisy function; and $N(0, \sigma^2)$ is normal, zero maen distribution with and deviation σ. For all benchmark functions the minima are at the origin or very close to the origin. Except for f_5 (Levy No. 5 function), its minima is at $\vec{x} = [-1.3068, 1.4248]$ with $f(\vec{x}) = -176.1375$.

- Sphere (50D)

$$f_1(x) = \sum_{i=1}^{D} {x_i}^2, \text{with} -100 \le x_i \le 100$$

- Rosenbrock (50D)

$$f_2(x) = \sum_{i=1}^{D} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \quad \text{with} -50 \le x_i \le 50$$

- Rastrigin (50D)

$$f_3(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10], \quad \text{with} -5.12 \le x_i \le 5.12$$

- Griewangk (50D)

$$f_4(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1, \quad \text{with} -600 \le x_i \le 600$$

- Levy No. 5 (2D)

$$f_5(x) = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \times \sum_{j=1}^{5} j\cos[(j+1)x_2 + j] + (x_1 + 1.42513)^2 + (x_1 + 0.80032)^2, \quad \text{with} -10 \le x_i \le 10$$

- Beale (2D)

$$f_6(x) = [1.5 - x_1(1 - x_2)]^2 + [2.25 - x_1(1 - x_2^2)]^2 + [2.625 - x_1(1 - x_2^3)]^2, \quad \text{with} -10 \leq x_i \leq 10$$

- Ackley (50D)

$$f_7(x) = -20e^{-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i^2}{D}}} - e^{\frac{\sum_{i=1}^{D} \cos(2\pi x_i)}{D}} + 20 + e, \quad \text{with} -32 \leq x_i \leq 32$$

- Schaffer's f_6 (2D)

$$f_8(x) = 0.5 - \frac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2}, \quad \text{with} -100 \leq x_i \leq 100$$

- De Jong's f_4 with noise (50D)

$$f_9(x) = \sum_{i=1}^{D} i x_i{}^4 + rand(0, 1), \quad \text{with} -1.28 \leq x_i \leq 1.28$$

C. Simulation Strategy

Like other studies in the evolutionary optimization field [4], [6], [17], for all conducted experiments, trials are repeated 50 times per function per noise deviation. Each run is continued up to 10^5 function calls and then mean and standard deviation of the best fitness values are reported. Re-sampling and thresholding techniques [5] are not applied in this paper.

In the last part of the conducted experiments, there is another contest to compare convergence speed and robustness of DE and ODE, settings and strategy for mentioned experiment are explained later.

D. Simulation Results

Numerical results are summarized in Tables II. These Tables show the results, mean $\pm$ (standard deviation), for each specified noise standard deviation (σ^2), 0 (noise-free), 0.25, 0.50, 0.75, and 1, respectively. The best result for each case is highlighted in **boldface**.

Performance comparison between DE and ODE is visualized in Fig. 2. Best fitness-value-so-far vs. number-of-function-calls is plotted for all benchmark functions. Again experiments have been repeated 50 times to plot average values. Because of space limitation, the plots only for noise deviation of $\sigma^2 = 0.5$ are given.

Furthermore, by using the numerical results summarized in Tables II, the plots of optimal-solution vs. noise-deviation ($\sigma^2 \in \{0, 0.25, 0.50, 0.75, 1\}$) for all benchmark functions are shown in Fig. 3.

In the second part of the conducted experiments, as we mentioned before, we compare the convergence speed and robustness of DE and ODE. The number of function calls (NFC) and the success rate (SR) are the meaningful metrics which are commonly used in the literature. By 50 times

TABLE II

MEAN $\pm$ (STANDARD DEVIATION) OF THE BEST FITNESS VALUE, FOR EACH SPECIFIED NOISE STANDARD DEVIATION (σ^2), 0 (NOISE-FREE), 0.25, 0.50, 0.75, AND 1, (TOP TO BOTTOM) RESPECTIVELY. THE BEST RESULT FOR EACH CASE IS HIGHLIGHTED IN **BOLDFACE**.

Function	DE	ODE
	$\sigma^2 = 0.0$	
f_1(50D)	$0.003 \pm (0.001)$	**$0.000 \pm (0.000)$**
f_2(50D)	$74.443 \pm (45.372)$	**$52.079 \pm (24.807)$**
f_3(50D)	$371.338 \pm (17.212)$	**$142.933 \pm (78.526)$**
f_4(50D)	$0.004 \pm (0.002)$	**$0.001 \pm (0.003)$**
f_5(2D)	$-176.138 \pm (0.000)$	$-176.138 \pm (0.000)$
f_6(2D)	$0.000 \pm (0.000)$	$0.000 \pm (0.000)$
f_7(50D)	$0.014 \pm (0.004)$	**$0.000 \pm (0.000)$**
f_8(2D)	$0.000 \pm (0.000)$	$0.000 \pm (0.000)$
f_9(50D)	$0.000 \pm (0.000)$	$0.000 \pm (0.000)$
	$\sigma^2 = 0.25$	
f_1(50D)	$0.520 \pm (0.100)$	**$0.417 \pm (0.107)$**
f_2(50D)	$77.930 \pm (49.683)$	**$57.325 \pm (27.978)$**
f_3(50D)	$373.788 \pm (15.616)$	**$149.580 \pm (70.987)$**
f_4(50D)	$1.428 \pm (0.106)$	**$1.399 \pm (0.111)$**
f_5(2D)	**$-176.108 \pm (0.025)$**	$-176.113 \pm (0.024)$
f_6(2D)	$0.033 \pm (0.032)$	**$0.023 \pm (0.020)$**
f_7(50D)	$1.944 \pm (0.345)$	**$0.994 \pm (0.243)$**
f_8(2D)	**$0.485 \pm (0.076)$**	$0.492 \pm (0.054)$
f_9(50D)	$0.201 \pm (0.078)$	**$0.153 \pm (0.074)$**
	$\sigma^2 = 0.5$	
f_1(50D)	$0.938 \pm (0.201)$	**$0.874 \pm (0.173)$**
f_2(50D)	$81.257 \pm (52.682)$	**$56.376 \pm (24.826)$**
f_3(50D)	$373.966 \pm (17.133)$	**$126.863 \pm (61.624)$**
f_4(50D)	$1.895 \pm (0.188)$	**$1.844 \pm (0.209)$**
f_5(2D)	$-176.086 \pm (0.067)$	**$-176.078 \pm (0.058)$**
f_6(2D)	$0.078 \pm (0.075)$	**$0.061 \pm (0.049)$**
f_7(50D)	$21.180 \pm (1.488)$	**$10.607 \pm (8.679)$**
f_8(2D)	$0.501 \pm (0.001)$	**$0.500 \pm (0.000)$**
f_9(50D)	$0.462 \pm (0.214)$	**$0.330 \pm (0.157)$**
	$\sigma^2 = 0.75$	
f_1(50D)	$1.496 \pm (0.358)$	**$1.227 \pm (0.340)$**
f_2(50D)	$77.029 \pm (50.721)$	**$57.045 \pm (26.143)$**
f_3(50D)	$374.726 \pm (15.447)$	**$139.681 \pm (71.997)$**
f_4(50D)	$2.332 \pm (0.358)$	**$2.185 \pm (0.317)$**
f_5(2D)	**$-176.045 \pm (0.094)$**	$-176.048 \pm (0.078)$
f_6(2D)	**$0.098 \pm (0.112)$**	$0.107 \pm (0.161)$
f_7(50D)	$21.602 \pm (0.224)$	**$17.884 \pm (6.548)$**
f_8(2D)	$0.500 \pm (0.000)$	**$0.499 \pm (0.006)$**
f_9(50D)	$0.616 \pm (0.226)$	**$0.438 \pm (0.202)$**
	$\sigma^2 = 1$	
f_1(50D)	$1.830 \pm (0.372)$	**$1.729 \pm (0.484)$**
f_2(50D)	$76.651 \pm (44.276)$	**$62.054 \pm (33.126)$**
f_3(50D)	$372.181 \pm (15.154)$	**$156.033 \pm (66.979)$**
f_4(50D)	$2.717 \pm (0.378)$	**$2.629 \pm (0.436)$**
f_5(2D)	$-176.030 \pm (0.118)$	**$-176.012 \pm (0.117)$**
f_6(2D)	$0.165 \pm (0.163)$	**$0.133 \pm (0.152)$**
f_7(50D)	$21.617 \pm (0.229)$	**$20.798 \pm (3.102)$**
f_8(2D)	$0.500 \pm (0.000)$	**$0.496 \pm (0.0203)$**
f_9(50D)	$0.890 \pm (0.356)$	**$0.718 \pm (0.338)$**

run, each time the DE and ODE try to reduce the function value below of VTR=0.01 (-176.3 for f_5) before meeting maximum number of function calls which is 3×10^5 for f_3, 2×10^5 for f_2 and f_7, and 10^5 for others. Number of times (out of 50), for which the algorithm (DE or ODE) succeeds to touch the VTR (value to reach) is measured as the success rate. Numerical results are summarized in Table III.

TABLE III
COMPARISON OF CONVERGENCE SPEED (NFC) AND SUCCESS RATES. AR: ACCELERATION RATE (ACCELERATED BY ODE), SR: SUCCESS RATE.

Function	DE	ODE	AR
f_1(50D)	$93628, 49$	**53842, 50**	42%
f_2(10D)	**61522, 50**	$105800, 41$	-72%
f_3(10D)	$323852, 36$	**79304, 46**	76%
f_4(50D)	$94150, 49$	**56411, 50**	40%
f_5(2D)	$4394, 50$	**4158, 50**	5%
f_6(2D)	$1392, 50$	**1133, 50**	19%
f_7(50D)	$104040, 50$	**62526, 50**	40%
f_8(2D)	$4154, 50$	**3667, 50**	12%
f_9(50D)	$38040, 50$	**13058, 50**	66%
$\sum_{i=1}^{9}$ NFC$_i$ =	725172	379899 $\rightarrow$	(47.6%)
Overall SR of DE= 48.2 (out of 50)			
Overall SR of ODE= 48.6 (out of 50)			

E. Results Analysis

For $\sigma^2 = 0$ (noise-free functions, Table II) DE and ODE have the same behavior for f_5(Levy No. 5), f_6(Beale), f_8(Schaffer's f_6), and f_9(De Jong's f_4) but over other 5 benchmark functions ODE outperforms DE.

For $\sigma^2 = 0.25$ f_6(Beale) and f_9(De Jong's f_4) are joined to those 5 functions and ODE surpasses DE over these 7 benchmark functions. When the noise derivation reaches to 0.5 or 1 for all 9 functions ODE performs absolutely better than DE.

For $\sigma^2 = 0.75$, DE outperforms ODE on functions f_5(Levy No. 5) and f_6(Beale); but for other 7 functions ODE surpasses the DE. As a conclusion, for noise-free functions they perform the same or the ODE (for 5 cases) outperforms DE.

For other noisy cases ODE at least over 7 benchmark functions (out of 9) performs better than DE. It means the ODE is generally doing better for noise-free and also noisy functions than the DE.

By increasing the noise deviation, optimal solutions of both DE and ODE worsen (most of time linearly and with the almost the same slope), see Fig. 3. Interestingly, both f_2(Rosenbrock) and f_3(Rastrigin) functions show stable behavior against the noise variation (Fig. 3.b and 3.c).

The ODE outperforms DE over 8 benchmark functions on the basis of the convergence speed (number of function calls), see Table III. The acceleration rate (improvement) is 47.6% in overall. The robustness (success rate here) is almost the same for both in overall.

VI. CONCLUDING REMARKS

The conventional DE was enhanced by utilizing opposition-based optimization concept in three levels, namely, population initialization, generation jumping, and local improvement of the population's best member. Our limited experiments confirmed that the proposed opposition-based differential evolution (ODE) algorithm performs better than the DE in terms of convergence speed over noisy and noise-free functions. Experiments with much comprehensive and complex test set is required to conclude strongly. Unlike other approaches in this field which are noise-oriented algorithms and perform worse for noise-free functions, the proposed ODE has a consistent performance for both noise-free and noisy cases.

In our algorithm and in all mentioned three levels, the opposite estimation (looking at the opposite side) introduces a second chance to improve our candidate solutions and also to support a dynamic behavior to follow up the optimal solution in the noisy environments.

The main drawback is that the ODE has introduced two more control parameters (Jr and F$'$). According to our experiences, achieved by testing ODE with other benchmark functions (not reported in this paper), setting Jr ≈ 0.3 and F$' \approx 0.1$ can provide satisfactory results in general. Further study is required to suggest more reliable empirical values.

Acknowledgement- The authors would like to thank Erik Jonasson (visiting scholar at the University of Waterloo, Canada) for conducting comprehensive experiments.

REFERENCES

[1] L.J. Fooel, A.J. Owens, M.J. Walsh, *Artificial Intelligence through Simulnted Evolution*, John Wiley & Sons, New York, 1966.

[2] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, New York: Addison-Wesley, 1989.

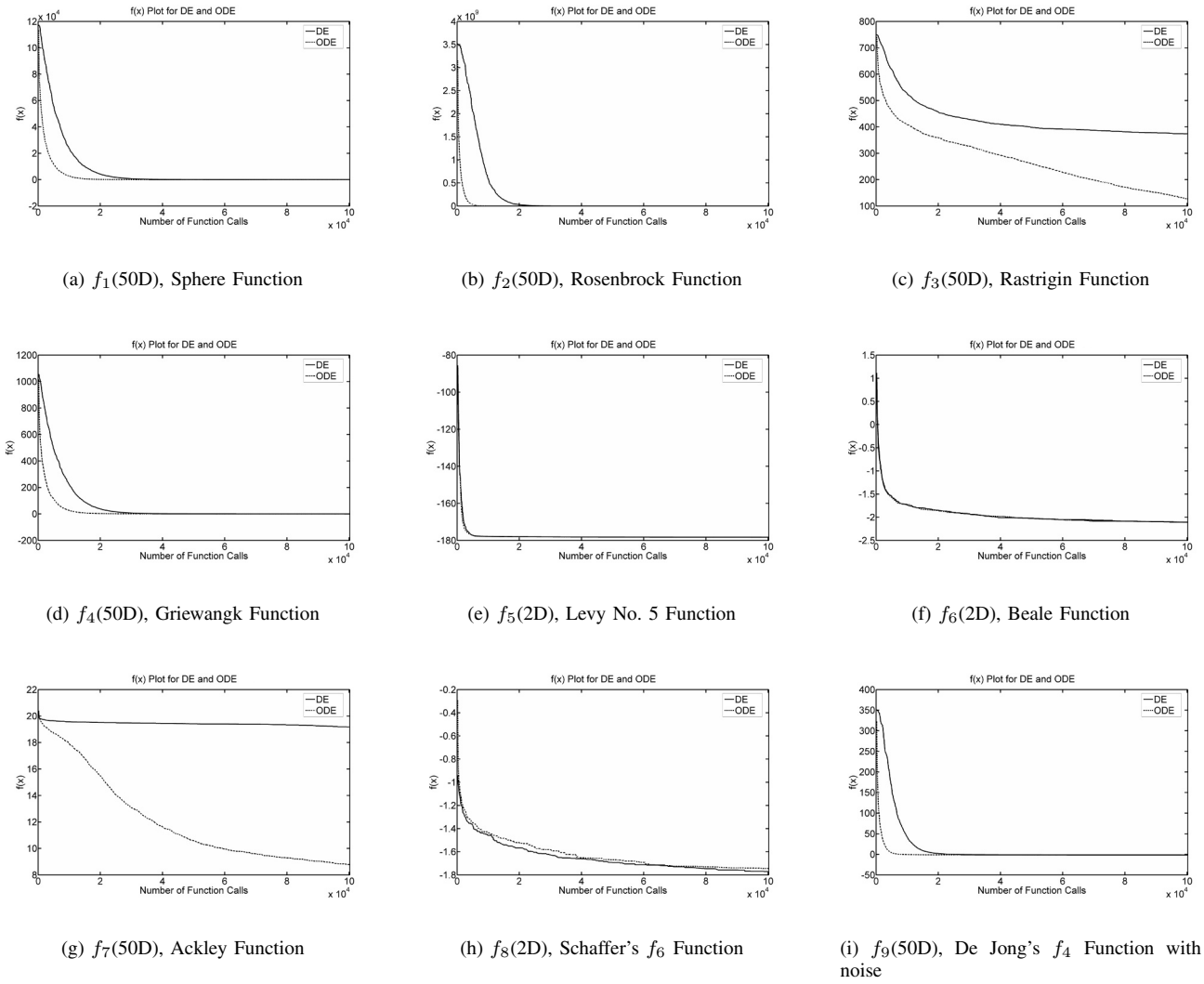
[3] J. Kennedyj and R.C. Eberhartr, *Particle Swarm Optimization*, Proceedings of the I995 IEEE International Conference on Neural Network (Perth, Australia, IEEE Service Center, Piscataway, NI), Vol. 4, pp. 1942-1948, 1995.

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

[5] Yaochu Jin and Jürgen Branke, *Evolutionary Optimization in Uncertain Environments- A Survey*, IEEE Transactions on Evolutionary Computatioon, Vol. 9, No. 3, pp. 303-317, June 2005.

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

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



(a) f_1(50D), Sphere Function

(b) f_2(50D), Rosenbrock Function

(c) f_3(50D), Rastrigin Function

(d) f_4(50D), Griewangk Function

(e) f_5(2D), Levy No. 5 Function

(f) f_6(2D), Beale Function

(g) f_7(50D), Ackley Function

(h) f_8(2D), Schaffer's f_6 Function

(i) f_9(50D), De Jong's f_4 Function with noise

Fig. 2. Performance comparison between DE and ODE. Best-fitness value-so-far vs. number-of-function-calls (NFC) is plotted for 9 benchmark functions (noise deviation: $\sigma^2 = 0.5$). Experiments have been repeated 50 times to plot by average values.

[8] H.R. Tizhoosh, *Reinforcement Learning Based on Actions and Opposite Actions.* ICGST International Conference on Artificial Intelligence and Machine Learning (AIML-05), Cairo, Egypt, 2005.

[9] H.R. Tizhoosh, *Opposition-Based Reinforcement Learning*, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 10, No. 3, 2006.

[10] K. Price, *An Introduction to Differential Evolution*, In: D. Corne, M. Dorigo, F. Glover (eds) New Ideas in Optimization, NcGraw-Hill, London (UK), pp. 79-108, 1999, ISBN:007-709506-5.

[11] Godfrey C. Onwubolu and B.V. Babu, *New Optimization Techniques in Engineering*, Berlin ; New York : Springer, 2004.

[12] R. Storn and K. Price, *Differential Evolution- A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*, Journal of Global OPtimization 11, pp. 341-359, 1997.

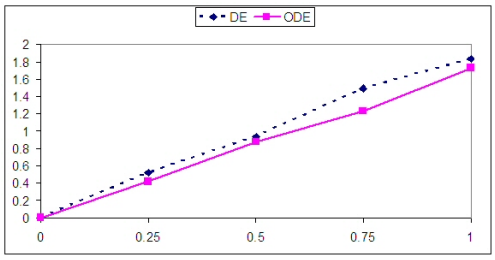
[13] X. Yao, Y. Liu, G. Lin, *Evolutionary Programming Made Faster*, IEEE Transactions on Evolutionary Computatioon, Vol. 3, No. 2, pp. 82-102, 1999.

[14] Thomas Back, *Evolutionary Algorithms in Theory and Practice : Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press, USA, 1996, ISBN: 0195099710.

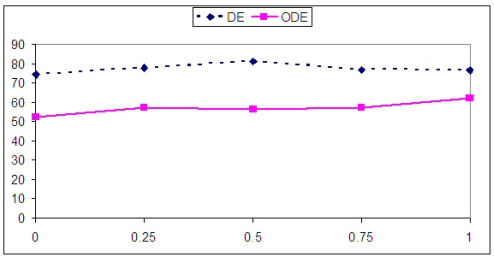
[15] A.E. Eiben and J.E. Smith, *Introduction to Evolutionary Computing (Natural Computing Series)*, Springer; 1st Edition, 2003, ISBN: 3540401849.

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

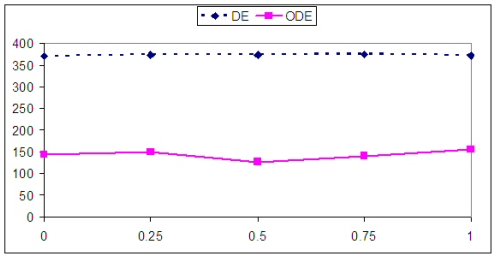
[17] J. Vesterstrøm and R. Thomsen, *A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems.* Proceedings of the Congress on Evolutionary Computation (CEC'04), IEEE Publications, Vol. 2, pp. 1980-1987, 2004.



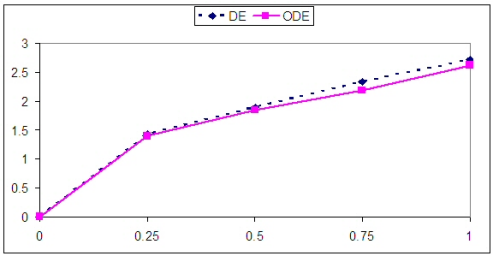
(a) f_1(50D), Sphere



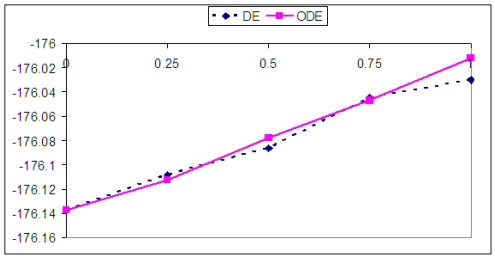
(b) f_2(50D), Rosenbrock Function



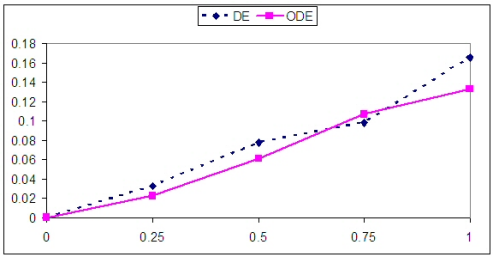
(c) f_3(50D), Rastrigin Function



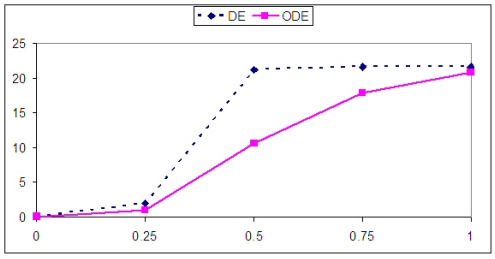
(d) f_4(50D), Griewangk Function



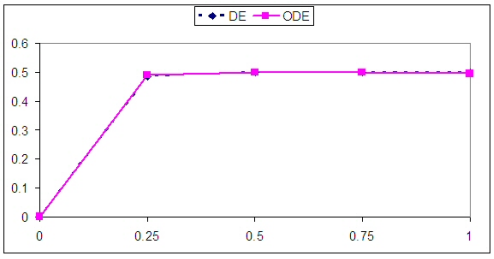
(e) f_5(2D), Levy No. 5 Function



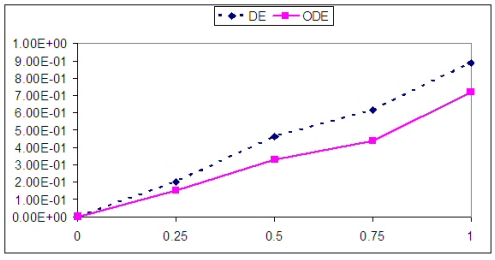
(f) f_6(2D), Beale Function



(g) f_7(50D), Ackley Function



(h) f_8(2D), Schaffer's f_6 Function



(i) f_9(50D), De Jong's f_4 Function with noise

Fig. 3. Plot of optimal-solution vs. noise-deviation ($\sigma^2 \in \{0, 0.25, 0.5, 0.75, 1\}$) for corresponding 9 benchmark functions.