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

Opposition-Based Differential Evolution Algorithms

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

***Abstract*— Evolutionary Algorithms (EAs) are well-known optimization approaches to cope with non-linear, complex problems. These population-based algorithms, however, suffer from a general weakness; they are computationally expensive due to slow nature of the evolutionary process. This paper presents some novel schemes to accelerate convergence of evolutionary algorithms. The proposed schemes employ opposition-based learning for population initialization and also for generation jumping. In order to investigate the performance of the proposed schemes, Differential Evolution (DE), an efficient and robust optimization method, has been used. The main idea is general and applicable to other population-based algorithms such as Genetic algorithms, Swarm Intelligence, and Ant Colonies. A set of test functions including unimodal and multimodal benchmark functions is employed for experimental verification. The details of proposed schemes and also conducted experiments are given. The results are highly promising.**

I. Introduction

Evolutionary algorithms (EAs) [1], [2] have been introduced to solve complex optimization problems. Some well-established and commonly used EAs are Genetic Algorithms (GA) [3] and Differential Evolution (DE) [4], [5]. Each of these method has its own characteristics, strengths, and weaknesses; but long computational time is a common drawback for both of them, specially when the solution space is hard to explore. Many efforts have been already done to accelerate convergence of these methods.

This paper presents a new scheme for evolutionary algorithms by applying opposition-based learning [6] to make EAs faster. 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. The idea is applicable to a wide range of optimization methods. Although the proposed schemes are embedded in a classical DE, but are general enough to be applied to all evolutionary algorithms.

Organization of this paper is as follows: In section II, the concept of opposition-based learning is explained. The proposed schemes are presented in section III. DE, our evolutionary testbed to implement the proposed schemes, is briefly reviewed in section IV. Experimental verifications are given in section V. Concluding remarks and future works form sections VI and VII, respectively. And finally, Appendix A describes characteristics of benchmark functions which have been employed in the conducted experiments.

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).

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 staring with opposite guess can accelerate convergence. The same approach can be applied not only to initial solutions but also to each solution in the current population. The concept of opposition-based learning was introduced in [6]. Applications were introduced in [6]–[8]. Before concentrating on opposition-based learning, we need to define opposite numbers [6]:

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

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

Similarly, this definition can be extended to higher dimensions as follows [6]:

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.

0-7803-9487-9/06/$20.00/©2006 IEEE

In the next section, the opposition-based optimization concept is employed to introduce new schemes of evolutionary algorithms, and to accelerate convergence rate.

III. Proposed Schemes

In this section, the concept of opposition-based optimization is applied to accelerate convergence of evolutionary algorithms. The main steps of evolutionary algorithms are shown in Fig. 1. As seen, after population initialization, algorithm remains inside a loop and continues to produce new generations (by applying selection, crossover, and mutation operations) and stops if termination criterion is satisfied. Initialization and producing new generations are two stages that can be extended by opposition-based concept. These two schemes will be introduced in following subsections.

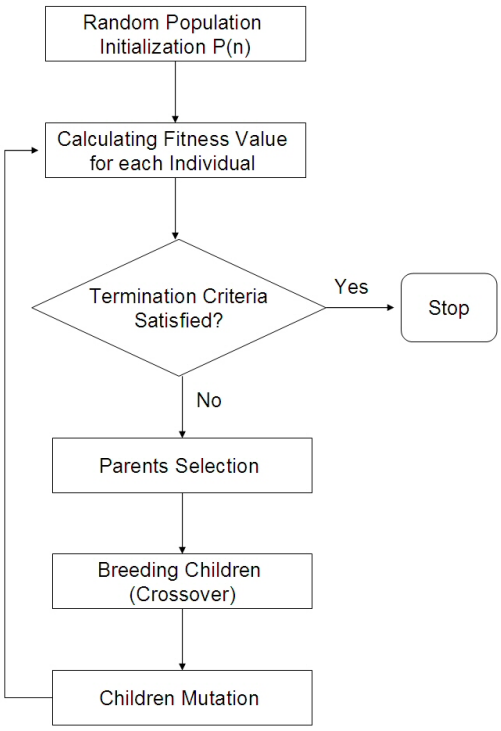


Fig. 1. Flowchart of evolutionary algorithms.

A. Opposition-Based Population Initialization

According to our review of optimization literature, random number generation is, in absence of a priori knowledge, 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. Many approaches can be proposed to generate better initial population based on opposition idea. One possible scheme is given in Fig. 2 and the corresponding algorithm is presented Table I. We call this variation ODE1 as an opposition-based extension of DE.

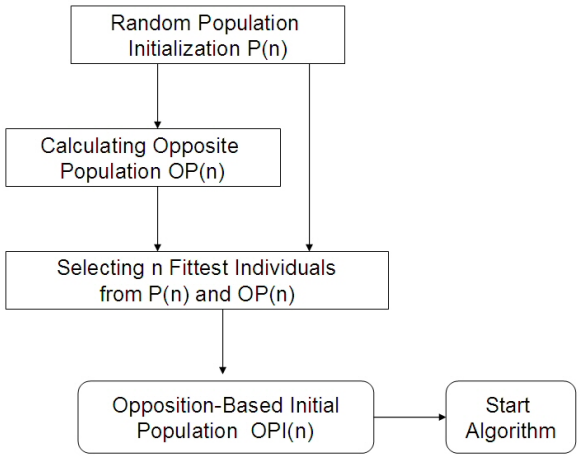


Fig. 2. Opposition-based population initialization for evolutionary algorithms, ODE1.

B. Opposition-Based Generation Jumping

By applying a similar approach to the current population, it can be forced to jump. Based on a jumping rate JR, instead of generating new population by selection, crossover, and mutation, the opposite population is calculated and the n fittest individuals are selected from the current population and the opposite population. Fig. 3 shows embedding of opposition-based population initialization and also generation jumping in the evolutionary algorithms. Table II presents corresponding algorithm, ODE2, the second variation of opposition-based DE. Our comprehensive experiments show that JR should be a small number ($\in (0, 0.4)$).

Dynamic Opposition: It should be noted here that in order to calculate the opposite population for generation jumping, the opposite of each variable is calculated dynamically. 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 ($[a_j, b_j]$):

$$OP_{k,j} = a^p{}_j + b^p{}_j - P_{k,j}, \qquad (3)$$
$$k = 1, 2, ..., n; j = 1, 2, ..., N_v.$$

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. For this reason, we calculate opposition point by using variables' current interval in the population ($[a^p{}_j, b^p{}_j]$) which is smaller than the corresponding initial range ($[a_j, b_j]$). In the following section, a short review of differential evolution approach, which we use as a case study to demonstrate embedding the opposition-based concept, is presented.

TABLE I

OPPOSITION-BASED POPULATION INITIALIZATION ALGORITHM, ODE1.

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 */

  Generating uniformly distributed random population; /* $P(n)$ */
  $OP_{k,j} = a_j + b_j - P_{k,j}$; /* calculating opposite population, $OP(n)$ */
  Selecting n fittest individuals from set the $\{P(n), OP(n)\}$ as initial population; /* $OPI(n)$ */
end

TABLE II

DE WITH EMBEDDED OPPOSITION-BASED POPULATION INITIALIZATION AND GENERATION JUMPING, ODE2.

begin
  n = population size;
  $\text{NFC}_{\text{MAX}=}$ maximum number of function calls;
  VTR = value to reach;
  JR = jumping rate;

  Opposition-Based Population Initialization; /* see Table I */
  Calculate Fitness Value for each Individual in the Population;
    while ($Best\_Fitness\_Value\_so\_far >$ VTR **and** NFC $<$ NFC_{MAX})
      if (rand (0,1) $<$ JR)
        /* Opposition-Based Jumping */
        $OP_{k,j} = a^p{}_j + b^p{}_j - P_{k,j}$; /* calculating opposite population of Current_Population, $OP(n)$ */
        Calculate Fitness Value for each Individual in $OP(n)$;
        Selecting n fittest individuals from $\{OP(n), \text{Current\_Population}\}$ as Current_Population;
      else
        /* DE evolution steps (mutation, crossover, and selection) */
        Mutation;
        Crossover;
        Selection;
      end if
      Calculate Fitness Value for each Individual in the Current_Population;
    end while
end

IV. A BRIEF INTRODUCTION TO DIFFERENTIAL EVOLUTION

Differential Evolution (DE) is a population-based, efficient, robust, and direct search method [9]. Like other evolutionary algorithms, it starts with an initial population vector, which is randomly generated when no preliminary solution is available. 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) [10]. 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 ($DE/rand/1/bin$), the mutation, crossover, and selection have straightforward procedures as follows [5], [10]:

Mutation - For each vector i from generation G a mutant vector $V_{i,G}$ is defined by

$$V_{i,G} = X_{r1,G} + F(X_{r2,G} - X_{r3,G}), \tag{4}$$

where $i = \{1, 2, ..., n\}$ and r_1, r_2, and r_3 are mutually different random integer indices selected from $\{1, 2, ..., n\}$.

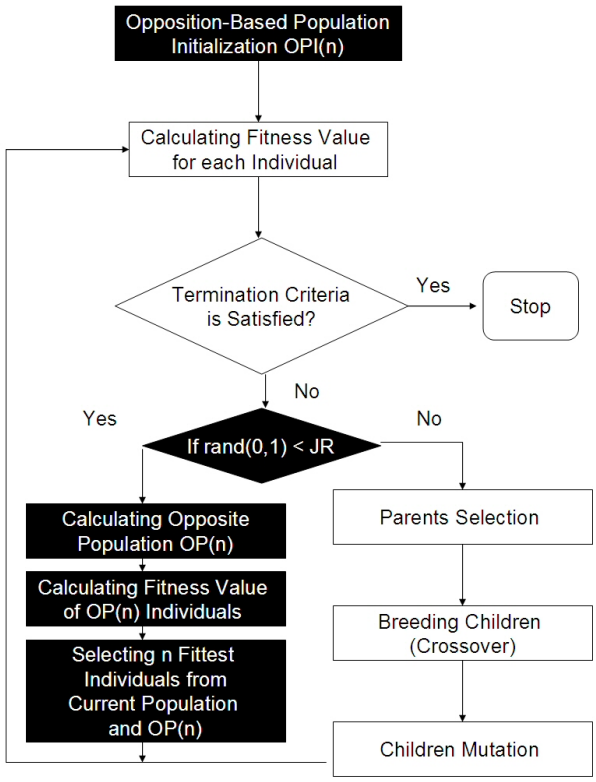


Fig. 3. Flowchart of embedding opposition-based population initialization and generation jumping for evolutionary algorithms: rand(0,1) is an evaluation of uniform random number ($\in (0,1)$) and JR is the jumping rate.

Further, i, r_1, r_2, and r_3 are different so $n \geq 4$. F is a real constant ($\in [0,2]$) 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.

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

$$U_{i,G} = (U_{1i,G}, U_{2i,G}, ..., U_{N_v i,G}), \tag{5}$$

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

$$U_{ji,G} = \begin{cases} V_{ji,G} & \text{if } rand_j(0,1) \leq Cr, \\ X_{ji,G} & \text{otherwise.} \end{cases} \tag{6}$$

Cr is predefined crossover constant $\in (0,1)$; $rand_j(0,1)$ is jth evaluation of uniform random generator $\in [0,1]$. Most popular value for Cr is in the range of $(0.4, 1)$ [11].

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 higher fitness value is chosen.

There are other variants of DE [5] but to maintain a general comparison, the classical version of DE, $DE/rand/1/bin$ [5], [10], has been selected to be investigated in conducted experiments.

V. Experimental Verification

The conducted experiments are categorized in three groups in order to investigate the performance of

- *Opposition-based initialization in general.*
- *DE with opposition-based initialization, ODE1 (see Table I).*
- *DE with embedded opposition-based initialization and generation jumping, ODE2 (see Table II).*

All experiments here have been repeated 100 times for each benchmark function to obtain statistically reliable performance numbers.

A. First Experimental Series

In this section, the possibility of starting with better initial population (population with lower average fitness value for minimization problems) is investigated. For each benchmark function (F1 to F7, see Appendix A) the random and opposition-based initial populations are generated and the average fitness value $\bar{f}$ of population is calculated. The results are summarized in Table III (population size: 100, benchmark functions' dimension: 10).

Last column of the Table III shows the achieved average fitness value improvement, $\bar{f}_{Imp}$, when the opposition-based approach (Table I) has been applied to generate initial population.

Now, one question arises: Can the average fitness value improvement still be achieved if dimensionality increases? In order to answer this question, experiments were repeated for higher dimensions 30, 60, 90, 150, and 300. The results are given in Fig. 4. As seen, by increasing the dimensionality the average fitness value improvement decreases, except for F3.

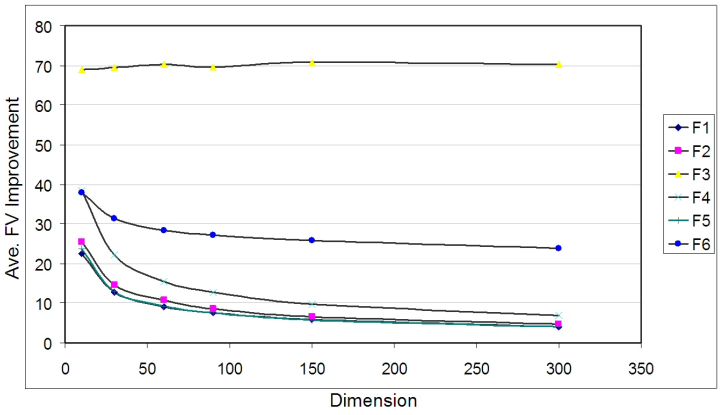


Fig. 4. Average fitness value improvement ($\bar{f}_{Imp}$) vs. dimension (D).

However, even for much higher dimensions, 300, improvement is still recognizable. Table IV presents these improvements, ranging form 4% to 70%. Function F7 is absent in this experiment because it is a one-dimensional function.

TABLE III

RANDOM AND OPPOSITION-BASED POPULATION INITIALIZATIONS.

$\bar{f}$: AVERAGE FITNESS VALUE OF POPULATION, σ: STANDARD DEVIATION, $\bar{f}_{Imp}$: AVERAGE FITNESS VALUE IMPROVEMENT. EXPERIMENTS HAVE BEEN REPEATED 100 TIMES TO CALCULATE AVERAGE VALUES.

	Random Initialization		**Opposition-Based Initialization**		
Function	$\bar{f}$	σ	$\bar{f}$	σ	$\bar{f}_{Imp}$
F1	$8.7465e+005$	$2.4543e+005$	$6.7861e+005$	$1.3419e+005$	$+22\%$
F2	$4.7800e+006$	$1.5255e+006$	$3.5576e+006$	$8.0131e+005$	$+26\%$
F3	$7.6707e+004$	$8.2063e+004$	$2.3847e+004$	$1.1196e+004$	$+69\%$
F4	$4.0464e+003$	$2.0179e+003$	$2.4895e+003$	825.7059	$+34\%$
F5	299.6832	85.0089	231.7410	46.8247	$+23\%$
F6	1.5915	0.7595	0.9886	0.3579	$+38\%$
F7	$1.1812e+005$	$2.0401e+005$	$1.1108e+003$	$1.9694e+003$	$+99\%$

TABLE IV

AVERAGE FITNESS VALUE IMPROVEMENT FOR $N = 300$. FUNCTION F7 IS ABSENT IN THIS EXPERIMENT BECAUSE IT IS A ONE-DIMENSIONAL FUNCTION.

Function	$\bar{f}_{Imp}$
F1	$+4\%$
F2	$+5\%$
F3	$+70\%$
F4	$+7\%$
F5	$+4\%$
F6	$+24\%$

The experiments were repeated with increased variable ranges. The results for almost all cases remained unchanged, with less than $\pm 2\%$ variation even for 100 times increase in range of variables.

Result Analysis - The conducted experiments in this section showed that by opposition-based population initialization, we can obtain an initial population which has lower average fitness value (for minimization problems) compared to random population. Increase in dimensionality or size of search space resulted in a drop of performance, however, the advantage of opposition-based initialization was still visible.

Next experiment series will show how this average fitness value improvement can make the convergence faster in an evolutionary algorithm such as DE.

B. Second Experimental Series

In this section, performance of Deferential Evolution (DE), with random population initialization, and DE with opposition-based population initialization (ODE1) are compared using a nine-function test suite (two harder to optimize functions, namely, Ackley's Path and Rastrigin are added to the previous test set. See Appendix A). For these experiments, like other works in this field [12], [13], the average number of function calls (NFC) and success rate (SR) have been used as performance measures. For these experiments and also for the next series the parameters are set as follows:

- Population size, $N_p = 100$
- Differential amplification factor, F= 0.5
- Crossover probability constant, Cr= 0.9
- Jumping rate constant, JR= 0.3 (applied to second experimental series)
- Strategy [5]: $DE/rand/1/bin$ (classical version of DE)
- Value to reach, VTR= 0.1 (except for F7 which is 10^{-7})
- Maximum function calls, $MAX_{NFC}=5\times10^5$ (10^6 for F9)
- Termination criterion: Distance between the best value found by algorithm and theoretical optimum should be less than 1% of the theoretical optimum value OR number of functions call pass MAX_{NFC}.

The results are shown in Table V. As seen, function calls improvement (NFC_{Imp}) is between 0% and 96%; the overall improvement for eight functions is 3.50% (96% improvement for F6 is excluded because improvement was exceptionally high). No improvement is achieved for F4, Rosenbrock's valley function. For this function convergence to global optimum is difficult because it is located inside a long, narrow, parabolic shaped, flat valley and opposite points can't help to improve convergence rate. Success rate for all cases is 100%. It means both algorithms could solve problems in all 100 runs.

Result Analysis - Results of this section showed that the opposition-based population initialization speeds up convergence. By applying opposition-based optimization at the initialization level, the classical DE has been made faster for our test set. In most optimization problems, which are solved by evolutionary algorithms, we can observe that the initial steps towards solution occur very fast. Most of the time is then spent to improve candidate results. As seen, even starting with better initial candidate solutions could not reduce the number of function calls so much. There are many optimization problems which each function call is time consuming, sometimes several hours (e.g. returning a simulation results as a fitness value) [10]. For these cases,

TABLE V

COMPARISON OF CLASSICAL DEFERENTIAL EVOLUTION WITH RANDOM POPULATION INITIALIZATION (DE) AND DE WITH OPPOSITION-BASED POPULATION INITIALIZATION (ODE1). D: DIMENSION, NFC: AVERAGE NUMBER OF FUNCTION CALLS, NFC_{IMP}: NUMBER OF FUNCTION CALLS IMPROVEMENT (ROUNDED). EXPERIMENTS HAVE BEEN REPEATED 100 TIMES TO CALCULATE AVERAGE VALUES.

Function	D	NFC (DE)	NFC (ODE1)	NFC_{Imp}
F1	60	113957	110928	+3%
F2	20	45553	44872	+1%
F3	20	77997	75617	+3%
F4	40	510490	510705	+0%
F5	20	47553	42370	+11%
F6	60	72974	2904	+96%
F7	1	2522	2407	+5%
F8	30	51666	50550	+2%
F9	20	635919	615250	+3%
Overall improvement of NFC for nine functions: 3.50% (F6 is excluded because improvement for that is exceptionally high.)				

even small improvement in number of function calls can be worthwhile.

C. Third Experimental Series

This experimental series investigate the effects of applying opposition-based initialization and generation jumping simultaneously (ODE2, Table II). All DE settings and performance measures are the same as in previous experiment series (JR=0.3). The results are summarized in Table VI. As shown, overall improvement of 40% for average number of function calls is achieved for 9 benchmark functions. Success rate for all cases is 100% except F9 (ODE2) which was 96%. Figure 5 shows performance comparisons between classical DE and ODE2 (because of space limitation just some samples are presented).

Result Analysis - By combining opposition-based initialization and generation jumping, an improvement of 40% is achieved. Again except for F4, for all benchmark functions the improvement is obtained.

VI. CONCLUDING REMARKS

In this paper, new schemes for evolutionary algorithms were proposed. First, the concept of opposition-based optimization was introduced and then it was employed to speed up convergence of evolutionary algorithms. Deferential Evolution was chosen as a sample evolutionary algorithm to implement proposed schemes. The idea of opposition-based optimization was used for population initialization and generation jumping in our experiments (ODE1, ODE2). The results demonstrated that the proposed schemes can accelerate convergence of DE algorithms.

Although the classical Deferential Evolutionary algorithm was used in all conducted experiments in this paper, the main idea is quite general and can be applied to other evolutionary algorithms such as Genetic algorithms.

VII. FUTURE WORKS

The proposed schemes have a high potential to improve performance of optimization methods and can be embedded in various steps of evolutionary algorithms. This work presents preliminary results along this path and demonstrates usefulness of the opposition-based optimization. Working on other possible schemes of the proposed idea, for instance, making smarter jumping, applying to other optimization methods, and also utilizing a more comprehensive test set are our directions for future works.

APPENDIX A. LIST OF BENCHMARK FUNCTIONS

All following functions are well-known benchmark functions which have been frequently used in literature [10], [14]. All of these functions are minimization problems.

- F1: 1^{st} De Jong function

$$f(x) = \sum_{i=1}^{D} {x_i}^2, \qquad -512 \leq x_i \leq 512$$

Global minimum: $x_i = 0, \quad f(x_i) = 0.$
Characteristics: Continuous, convex, unimodal.

- F2: Axis parallel hyper-ellipsoid

$$f(x) = \sum_{i=1}^{D} i{x_i}^2, \qquad -512 \leq x_i \leq 512$$

Global minimum: $x_i = 0, \quad f(x_i) = 0.$
Characteristics: Continuous, convex, unimodal.

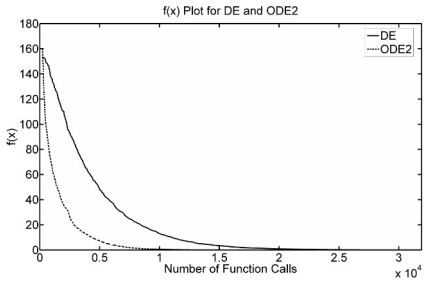
- F3: Rotated hyper-ellipsoid function

$$f(x) = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2, \qquad -65 \leq x_i \leq 65$$

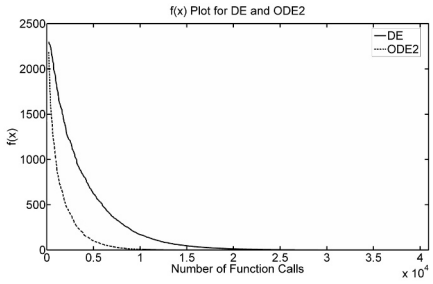
TABLE VI

CONVERGENCE COMPARISON OF DE AND ODE2 (OPPOSITION-BASED INITIALIZATION AND GENERATION JUMPING). EXPERIMENTS HAVE BEEN REPEATED 100 TIMES TO CALCULATE AVERAGE VALUES.

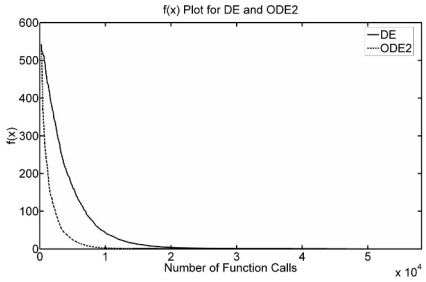
Function	D	NFC (DE)	NFC (ODE2)	NFC_{Imp}
F1	30	27938	13062	$+53\%$
F2	30	37528	18224	$+51\%$
F3	20	77502	73122	$+6\%$
F4	10	296260	366990	-24%
F5	30	53610	31242	$+42\%$
F6	30	6122	872	$+86\%$
F7	1	2910	2134	$+27\%$
F8	30	52556	28528	$+46\%$
F9	10	327782	92060	$+72\%$
Overall improvement of NFC for nine benchmark functions: 40%				



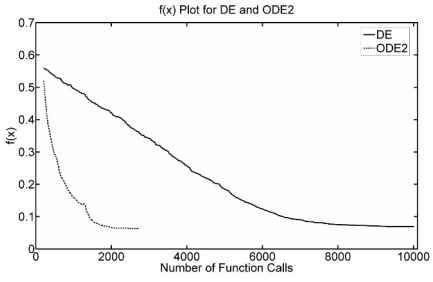
(a) F1, 53% faster



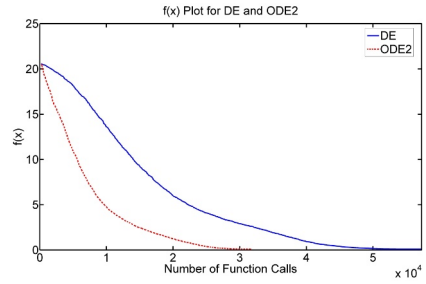
(b) F2, 51% faster



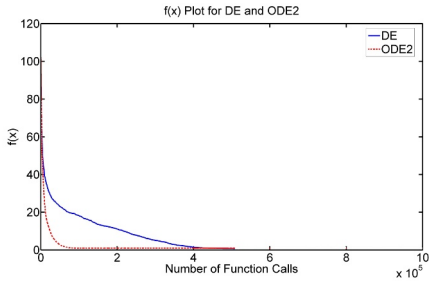
(c) F5, 42% faster



(d) F6, 86% faster



(e) F8, 46% faster



(f) F9, 72% faster

Fig. 5. Performance comparison between classical DE and ODE2. Progress toward optimum value (minimization, $f(x) = 0$). Experiments have been repeated 100 times to plot by average values.

Global minimum: $x_i = 0, \quad f(x_i) = 0.$
Characteristics: Continuous, convex, unimodal.

• F4: Rosenbrock's valley

$$f(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2], \ -2 \leq x_i \leq 2$$

Global minimum: $x_i = 1, \quad f(x_i) = 0.$
Characteristics: Convergence to global optimum is difficult because it is inside a long, narrow, parabolic shaped flat valley.

• F5: Griewangk function

$$f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1, \ -600 \leq x_i \leq 600$$

Global minimum: $x_i = 0, \quad f(x_i) = 0.$
Characteristics: Many regularly distributed local minima.

• F6: Sum of different power

$$f(x) = \sum_{i=1}^{D} |x_i|^{(i+1)}, \quad -1 \leq x_i \leq 1$$

Global minimum: $x_i = 0, \quad f(x_i) = 0.$
Characteristics: Continuous, convex, unimodal.

• F7: One dimensional multimodal function

$$f(x) = x^6 - 15x^4 + 27x^2 + 243, \quad -10 \leq x_i \leq 10$$

Global minimums: $x_i = +3/-3, \quad f(x_i) = 0.$
Characteristics: Continuous, multimodal.

• F8: Ackley's Path

$$f(x) = -20e^{-0.2\sqrt{\frac{\sum_{i=1}^{D} x_i^2}{D}}} - e^{\frac{\sum_{i=1}^{D} \cos(2\pi x_i)}{D}} + 20 + e,$$

$$-30 \leq x_i \leq 30$$

Global minimum: $x_i = 0, \quad f(x_i) = 0.$
Characteristics: Quite narrow attraction basin of the global minimum, multimodal.

• F9: Rastrigin function

$$f(x) = 10D + \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i)), \ -5.12 \leq x_i \leq 5.12$$

Global minimum: $x_i = 0, \quad f(x_i) = 0.$
Characteristics: Highly multimodal, regularly distributed local minima.

REFERENCES

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

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

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

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

[12] J. Andre, P. Siarry, T. Dognon, *An Improvement of the Standard Genetic Algorithm Fighting Premature Convergence in Continuous Optimization*, Advance in Engineering Software 32, pp. 49-60, 2001.

[13] Ondřej Hrstka and Anna Kučerová, *Improvement of Real Coded Genetic ALgorithm Based on Differential Operators Preventing Premature Convergence*, Advance in Engineering Software 35, pp. 237-246, 2004.

[14] 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.