Solving Large Scale Optimization Problems by Opposition-Based Differential Evolution (ODE)

Shahryar Rahnamayan
University of Ontario Institute of Technology (UOIT)
Electrical and Computer Engineering
2000 Simcoe Street North
Oshawa, ON, L1H 7K4, Canada
Shahryar.Rahnamayan@uoit.ca

G. Gary Wang
Simon Fraser University (SFU)
Mechatronic Systems Engineering
250-13450 102 Avenue
Surrey, BC, V3T 0A3, Canada
Gary_Wang@sfu.ca

Abstract: This work investigates the performance of Differential Evolution (DE) and its opposition-based version (ODE) on large scale optimization problems. Opposition-based differential evolution (ODE) has been proposed based on DE; it employs opposition-based population initialization and generation jumping to accelerate convergence speed. ODE shows promising results in terms of convergence rate, robustness, and solution accuracy. A recently proposed seven-function benchmark test suite for the CEC-2008 special session and competition on large scale global optimization has been utilized for the current investigation. Results interestingly confirm that ODE outperforms its parent algorithm (DE) on all high dimensional (500D and 1000D) benchmark functions (F_1-F_7). Furthermore, authors recommend to utilize ODE for more complex search spaces as well. Because results confirm that ODE performs much better than DE when the dimensionality of the problems is increased from 500D to 1000D. All required details about the testing platform, comparison methodology, and also achieved results are provided.

Key–Words: Opposition-Based Differential Evolution (ODE), Opposition-Based Optimization (OBO), Opposition-Based Computation (OBC), Cooperative Coevolutionary Algorithms (CCA), Large Scale Optimization, Scalability, High-Dimensional Problems

1 Introduction

Generally speaking, evolutionary algorithms (EAs) are well-established techniques to approach those practical problems which are difficult to solve for the classical optimization methods. Tackling problems with mixed-type of variables, many local optima, undifferentiable or non-analytical functions, which are frequently faced in all science and engineering fields, are some examples to highlight the outstanding capabilities of the evolutionary algorithms. Because of evolutionary nature of EA algorithms, as a disadvantage, they are computationally expensive in general [16, 17]. Furthermore, the performance of EAs decreases sharply by increasing the dimensionality of optimization problems. The main reason for that is increasing the search space dimensionality would increase complexity of the problem exponentially. On the other hand, for many real-world applications, we are faced with problems which contain a huge number of variables. Due to such a need, supporting the scalability is a very valuable characteristic for any optimization method. In fact, reducing the required number of function calls to achieve a satisfactory solution (which means accelerating convergence rate) is always valuable; especially when we are faced with expensive optimization problems. Employing smart sampling and meta-modelling are some commonly used approaches [24, 25] to tackle this kind of problems.

Many comparison studies confirm that the differential evolution (DE) outperforms many other evolutionary optimization methods. In order to enhance DE, opposition-based differential evolution (ODE) was proposed by Rahnamayan et al. in 2006 [2, 3, 5, 27] and then quasi-oppositional DE (QODE) in 2007 [4]. These algorithms (ODE and QODE) are based on DE and the opposition concept [8, 1]. ODE was followed by others to propose opposition-based particle swarm algorithms [20, 21], tracking dynamic objects using ODE [9], opposition-based ant colony algorithms [10, 11], enhancing self-adaptive DE with population size reduction to tackle large scale problems [19][1], and introducing an adaptive DE applied to tuning of a Chess program [22].

ODE employs opposition-based population initialization [6] and generation jumping to accelerate the convergence rate of DE. The main idea behind the opposition is the simultaneous consideration of an estimate and its corresponding opposite estimate (i.e., guess and opposite guess) in order to achieve a better approximation for the current candidate solution [29].

The reported results for ODE were promising on low and medium size problems ($D < 100$). But previously, ODE was not investigated in scalability. By experimental verification, current work tries to find out an answer for this question: which one, DE or ODE, presents higher efficiency to solve large scale problems?

Organization of this paper is as follows: Section 2 provides the brief overview of DE and ODE. In Section 3, detailed experimental results and also performance analysis are given and explained. Finally, the work is concluded in Section 4.

2 Brief Review of DE and ODE

Differential evolution (DE) and its extended version by opposition-based concept (ODE) has been briefly reviewed in following subsections.

2.1 Differential Evolution (DE)

Differential Evolution (DE) was proposed by Price and Storn in 1995 [12]. It is an effective, robust, and simple global optimization algorithm [13]. DE is a population-based directed search method [14]. Like other evolutionary algorithms, it starts with an initial population vector, which is randomly generated when no preliminary knowledge about the solution space is available. Each vector of the initial population can be generated as follows [13]:

$$X_{i,j} = a_j + rand_j(0,1) \times (a_j - b_j); j = 1, 2, ..., D, \tag{1}$$

where D is the problem dimension; a_j and b_j are the lower and the upper boundaries of the variable j, respectively. $rand(0,1)$ is the uniformly generated random number in $[0,1]$.

Let us assume that $X_{i,G}(i = 1, 2, ..., N_p)$ are candidate solution vectors in generation G (N_p : population size). Successive populations are generated by adding the weighted difference of two randomly selected vectors to a third randomly selected vector. For classical DE ($DE/rand/1/bin$), the mutation, crossover, and selection operators are straightforwardly defined as follows:

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

$$V_{i,G} = X_{a,G} + F(X_{c,G} - X_{b,G}), \tag{2}$$

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

Crossover - DE utilizes the crossover operation to generate new solutions by shuffling competing vectors and also to increase the diversity of the population. For the classical DE ($DE/rand/1/bin$), the binary crossover (shown by 'bin' in the notation) is utilized. It defines the following trial vector:

$$U_{i,G} = (U_{1i,G}, U_{2i,G}, ..., U_{Di,G}), \tag{3}$$

$$U_{ji,G} = \begin{cases} V_{ji,G} & \text{if } rand_j(0,1) \leq C_r \vee j = k, \\ X_{ji,G} & \text{otherwise.} \end{cases} \tag{4}$$

$C_r \in (0,1)$ is the predefined crossover rate, and $rand_j(0,1)$ is the j^{th} evaluation of a uniform random number generator. $k \in \{1, 2, ..., D\}$ is a random parameter index, chosen once for each i to make sure that at least one parameter is always selected from the mutated vector, $V_{ji,G}$. Most popular values for C_r are in the range of $(0.4, 1)$ [15].

Selection - This is an approach which must decide which vector ($U_{i,G}$ or $X_{i,G}$) should be a member of next (new) generation, $G + 1$. For a minimization problem, the vector with the lower value of objective function is chosen (greedy selection).

This evolutionary cycle (i.e., mutation, crossover, and selection) is repeated N_p (population size) times

[1]It uses opposition concept implicitly by changing the sign of F and so searching in the opposite direction.

to generate a new population. These successive generations are produced until meeting the predefined termination criteria.

2.2 Opposition-Based DE (ODE)

Similar to all population-based optimization algorithms, two main steps are distinguishable for the DE, population initialization and producing new generations by evolutionary operations such as selection, crossover, and mutation. ODE enhances these two steps based on looking at the opposite points (let say individuals in the population). The opposite point has a straightforward definition as follows:

Definition (Opposite Number) - Let $x \in [a, b]$ be a real number. The opposite number $\breve{x}$ is defined by

$$\breve{x} = a + b - x. \tag{5}$$

Similarly, this definition can be extended to higher dimensions as follows [8, 1, 29]:

Definition (Opposite Point in n-Dimensional Space) - 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 $\breve{P} = (\breve{x}_1, \breve{x}_2, ..., \breve{x}_n)$ is completely defined by its components

$$\breve{x}_i = a_i + b_i - x_i. \tag{6}$$

Fig.1 presents the flowchart of ODE. White boxes present steps of the classical DE and grey ones are expended by opposition concept. Blocks (1) and (2) present opposition-based initialization and opposition-based generation jumping, respectively.

Extended blocks by opposition concept will be explained in the following subsections.

2.2.1 Opposition-Based Population Initialization

By utilizing opposite points, we can obtain fitter starting candidate solutions even when there is no a priori knowledge about the solution(s). Block (1) in Fig.1 show implementation of corresponding opposition-based initialization for the ODE. Following steps show that procedure:

1. Random initialization of population $P(N_P)$,
2. Calculate opposite population by

$$OP_{i,j} = \mathrm{a}_j + \mathrm{b}_j - \mathrm{P}_{i,j}, \tag{7}$$

$$i = 1, 2, ..., N_p\ ;\ j = 1, 2, ..., D,$$

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

3. Selecting the N_p fittest individuals from $\{P \cup OP\}$ as initial population.

2.2.2 Opposition-Based Generation Jumping

By applying a similar approach to the current population, the evolutionary process can be forced to jump to a new solution candidate, which may be fitter than the current one. Based on a jumping rate Jr, after generating new population by selection, crossover, and mutation, the opposite population is calculated 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. The maximum and minimum values of each variable in *current population* ($[\mathrm{MIN}_j^p, \mathrm{MAX}_j^p]$) are used to calculate opposite points instead of using variables' predefined interval boundaries ($[a_j, b_j]$):

$$\mathrm{OP}_{i,j} = \mathrm{MIN}_j^p + \mathrm{MAX}_j^p - \mathrm{P}_{i,j}, \tag{8}$$

$$i = 1, 2, ..., N_p\ ;\ j = 1, 2, ..., D.$$

The dynamic opposition increases the chance to find fitter opposite points, so it helps in fine tuning. By staying within variables' interval static boundaries, we would jump outside of the shrunken solution space and the knowledge of current reduced space (converged population) would not be utilized. 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 (2) in Fig.1 shows the implementation of opposition-based generation jumping for the ODE. Our extensive experiments show that jumping rate Jr should be a small number in $(0, 0.4]$.

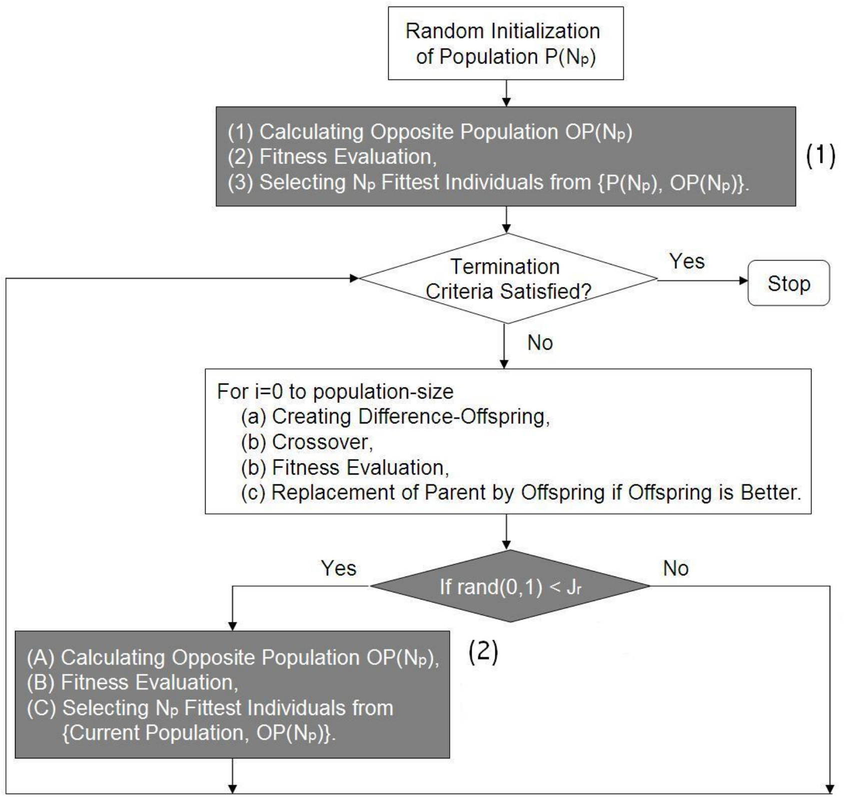

Random Initialization of Population P(Np)
(1) Calculating Opposite Population OP(Np)
(2) Fitness Evaluation,
(3) Selecting Np Fittest Individuals from {P(Np), OP(Np)}.
(1)
Termination Criteria Satisfied?
Yes
Stop
No
For i=0 to population-size
(a) Creating Difference-Offspring,
(b) Crossover,
(b) Fitness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

Figure 1: Opposition-Based Differential Evolution (ODE). Block (1): Opposition-based initialization, Block (2): Opposition-based generation jumping (Jr: jumping rate, $rand(0,1)$: uniformly generated random number, N_p: population size).

3 ODE vs. DE on Large Scale Minimization Problems

In this section, DE and ODE are compared on a large scale (D=500 and D=1000) minimization test suite in term of solution accuracy. The utilized test suite contains seven well-known unimodal and multi-modal functions with separability and non-separability characteristics in both modality groups.

3.1 Benchmark Functions

For comparison of DE and ODE, a recently proposed benchmark test suite for the CEC-2008 Special Session and Competition on Large Scale Global Optimization [23] has been utilized. It includes two unimodal (F_1-F_2) and five multi-modal (F_3-F_7) functions, among which four of them are non-separable (F_2, F_3, F_5, F_7) and three are separable (F_1, F_4, F_6). Functions names and their properties are summarized in Table 1. The mathematical definitions of these functions are described in Appendix A.

3.2 Parameter Settings

Parameter setting for all conducted experiments is as follows:

- Dimension of the problems, $D = 500$ and $D = 1000$ [23]
- Population size, $N_p = D$ [26, 19]
- Differential amplification factor, $F = 0.5$ [5, 7, 28]
- Crossover probability constant, $Cr = 0.9$ [5, 7, 28]
- Mutation strategy: DE/rand/1/bin (classical version of DE) [12, 5, 7, 28]
- Maximum number of function calls, $\text{MAX}_{\text{NFC}} = 5000 \times D$ [23]

Table 1: Benchmark functions. All of them are scalable and shifted.

Function	Name	Properties	Search Space
F_1	Shifted Sphere Function	Unimodal, Separable	$[-100, 100]^D$
F_2	Shifted Schwefels Problem 2.21	Unimodal, Non-separable	$[-100, 100]^D$
F_3	Shifted Rosenbrocks Function	Multi-modal, Non-separable, A narrow valley from local optimum to global optimum	$[-100, 100]^D$
F_4	Shifted Rastrigins Function	Multi-modal, Separable, Huge number of local optima	$[-5, 5]^D$
F_5	Shifted Griewanks Function	Multi-modal, Non-separable	$[-600, 600]^D$
F_6	Shifted Ackleys Function	Multi-modal, Separable	$[-32, 32]^D$
F_7	FastFractal DoubleDip Function	Multi-modal, Non-separable	$[-1, 1]^D$

- Jumping rate constant (for ODE), $Jr = 0.3$ [5, 7, 28]

All above mentioned settings are based on our or colleagues' previous works and so there has no new attempts to obtain better values for them. In order to maintain a reliable and fair comparison, these settings are kept unchanged for all conducted experiments for both algorithms and also for both dimensions (D=500 and 1000).

3.3 Comparison Criteria

The conducted comparisons in this paper are based on solution accuracy. The termination criteria is set to reaching the maximum number of function calls ($5000 \times D$). In order to have a clear vision on algorithm's efficiency, the best, median, worse, mean, standard deviation, and 95% confidential interval (95% CI) [2] of the error value ($f(x) - f(x^*)$, x^*: optimum vector) are computed with respect to 25 runs per function.

3.4 Numerical Results

Results for DE and ODE on seven functions are summarized in Table 2 for 500D and in Table 3 for 1000D. For each function, the best, median, worse, mean, standard deviation, and 95% confidential interval (95% CI) of the error value on 25 runs are presented. The best result of each error measure is emphasized in **boldface**. Fitness plots of DE and ODE for D=500 and D=1000 are given in Figure 2 and Figure 3, respectively. The plots show that how ODE converges to the solution faster than DE.

[2]It shows that 95% of the data appearances in this interval.

3.5 Result Analysis

As seen from Table 2 and Table 3, on all benchmark test functions, ODE clearly outperforms DE. Although, for functions F_2 (the only for 500D), F_4, F_6, and F_7, DE presents a lower standard deviation, the fact that even for these functions it is reported 95% confidential intervals confirms that ODE performs better. In fact, the smaller boundaries of 95% CI for ODE demonstrate this conclusion. That is valuable to mention, except for F_6, on all functions (D=500 and 1000), a big difference between DE and ODE's results is recognizable.

As mentioned before, our test suite contains shifted unimodal, multi-modal (with huge number of optima), scalable, separable, and non-separable functions; so according to the obtained results that is reasonable to say ODE presents evidences to perform better than DE (parent algorithm) on large scale problems.

4 Conclusion

Before the current work, the performance of ODE on large scale problems has not been investigated. So, it was interesting to have a performance study by an accepted high dimensional test suite. The achieved results are promising because ODE outperforms DE on all seven test functions, for D=500 and 1000. We propose that other DE-based approaches, which are used to tackle large scale problems, may investigate replacing DE by ODE.

Proposing a cooperative coevolutionary ODE (CCODE) and also studying ODE's jumping rate for large scale optimization represent our directions for future work.

Acknowledgement: Authors would like to thank Dr. K. Tang et al. who shared the Matlab code of benchmark functions.

References:

[1] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Opposition versus Randomness in Soft Computing Techniques*, Elsevier Journal on Applied Soft Computing, Volume 8, March 2008, pp. 906-918.

[2] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, *Opposition-Based Differential Evolution Algorithms*, IEEE World Congress on Computational Intelligence, Vancouver, Canada, 2006, pp. 7363–7370.

[3] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, *Opposition-Based Differential Evolution for Optimization of Noisy Problems*, IEEE World Congress on Computational Intelligence, Vancouver, Canada, 2006, pp. 6756–6763.

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

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

[6] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *A Novel Population Initialization Method for Accelerating Evolutionary Algorithms*, Elsevier Journal on Computers and Mathematics with Applications, Volume 53, Issue 10, May 2007, pp. 1605-1614.

[7] S. Rahnamayan, *Opposition-Based Differential Evolution*, Ph.D. Thesis, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, May 2007.

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

[9] Fadi Salama, *Tracking Dynamic Objects using Opposition-Based Differential,* Thesis Thesis for Honours Programme, Department of Computer Science, University of Western Australia, Australia, 2007.

[10] A.R. Malisia and H.R. Tizhoosh, *Applying Opposition-Based Ideas to the Ant Colony System*, Proceedings of IEEE Swarm Intelligence Symposium (SIS-2007), Hawaii, April 1-5, 2007, pp. 182-189.

[11] Alice R. Malisia, *Investigating the Application of Opposition-Based Ideas to Ant Algorithms*, M.Sc. Thesis, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario, Canada, Sep. 2007.

[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] 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.

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

[15] 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.

[16] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Learning Robust Object Segmentation from User-Prepared Samples,* WSEAS Transactions on Computers, Volume 4, Issue 9, Sep. 2005, pp. 1163-1170.

[17] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Towards Incomplete Object Recognition,* WSEAS, Transactions on Systems, Volume 4, Issue 10, October 2005, pp. 1725-1732.

[18] G. L. Cascella, F. Neri, N. Salvatore, G. Acciani, F. Cupertino, *Hybrid EAs for Backup Sensorless Control of PMSM Drives,* WSEAS

Transactions on Systems, Volume 5, Issue 1, pp. 131–135, January 2006.

[19] J. Bresta, A. Zamuda, B. Bošković, M.S. Maučec, V. Žumer, *High-Dimensional Real-Parameter Optimization using Self-Adaptive Differential Evolution Algorithm with Population Size Reduction*, IEEE World Congress on Computational Intelligence (WCCI-2008), Hong Kong, June 2008, pp. 2032-2039.

[20] L. Han, X. He, *A Novel Opposition-Based Particle Swarm Optimization for Noisy Problems*, Third International Conference on Natural Computation (ICNC-2007), 2007, pp. 624-629.

[21] H. Wang, Y. Liu, S. Zeng, H. Li, C. Li, *Opposition-based Particle Swarm Algorithm with Cauchy Mutation*, IEEE Congress on Evolutionary Computation (CEC-2007), Singapore, Sep. 2007, pp. 4750-4756.

[22] B. Bošković, S. Greiner, J. Brest, A. Zamuda, and V. Žumer, *An Adaptive Differential Evolution Algorithm with Opposition-Based Mechanisms, Applied to the Tuning of a Chess Program*, In Uday K. Chakraborty, editor, Advances in Differential Evolution, Studies in Computational Intelligence, Vol. 143, Springer, June 2008.

[23] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, Z. Yang, *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*, Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, http://nical.ustc.edu.cn/cec08ss.php, 2007.

[24] B. Sharif, G.G. Wang, and T. El-Mekkawy, *Mode Pursing Sampling Method for Discrete Variable Optimization on Expensive Black-box Functions*, ASME Transactions, Journal of Mechanical Design, Vol. 130, 2008, pp.021402-1-11.

[25] L. Wang, S. Shan, and G.G. Wang, *Mode-Pursuing Sampling Method for Global Optimization on Expensive Black-box Functions*, Journal of Engineering Optimization, Vol. 36, No. 4, August 2004, pp. 419-438.

[26] Z. Yang, K. Tang, X. Yao, *Differential Evolution for High-Dimensional Function Optimization*, IEEE Congress on Evolutionary Computation (CEC-2007), Singapore, Sep. 2007, 3523-3530.

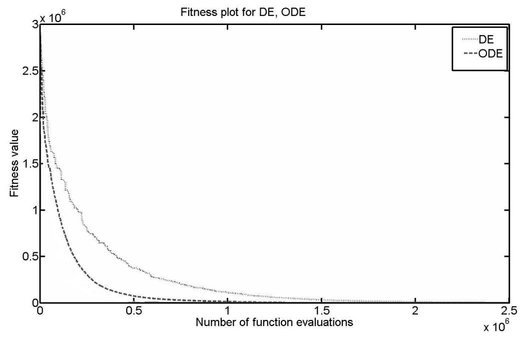
[27] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Opposition-Based Differential Evolution*, Advances in Differential Evolution, Series: Studies in Computational Intelligence, Springer-Verlag, 2008, ISBN: 978-3-540-68827-3, pp. 155-171.

[28] S. Rahnamayan, H.R. Tizhoosh, M.M.A Salama, *Differential Evolution via Exploiting Opposite Populations*, Oppositional Concepts in Computational Intelligence, Series: Studies in Computational Intelligence, Springer-Verlag, 2008, ISBN: 978-3-540-70826-1, pp. 143-160.

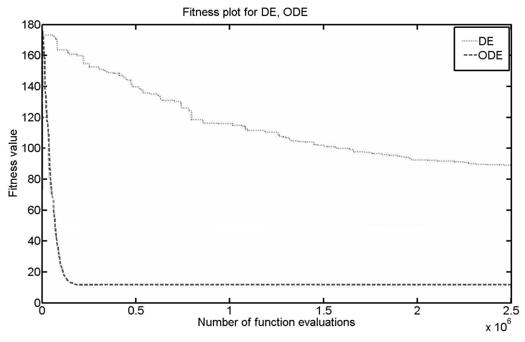
[29] H.R. Tizhoosh, M. Ventresca, S. Rahnamayan, *Opposition-Based Computing*, Oppositional Concepts in Computational Intelligence, Series: Studies in Computational Intelligence, Springer-Verlag, 2008, ISBN: 978-3-540-70826-1, pp. 11-28.

Table 2: Numerical results for DE and ODE on seven 500-dimensional minimization problems (25 runs per function). The best result of each error measure is emphasized in boldface. 95% CI stands for 95% confidential interval.

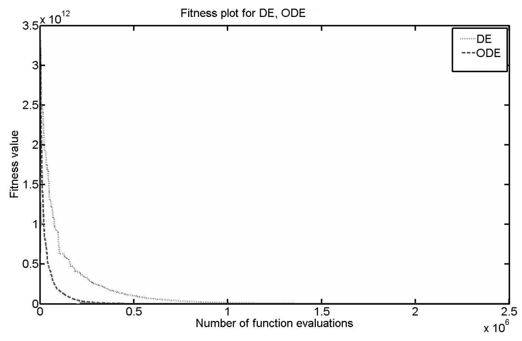
Function	Error Value	DE	ODE
F_1	Best	$2,636.54$	$\mathbf{15.66}$
	Median	$3,181.45$	$\mathbf{36.61}$
	Worse	$4,328.80$	$\mathbf{292.65}$
	Mean	$3,266.24$	$\mathbf{80.17}$
	Std	409.68	$\mathbf{79.24}$
	95% CI	$[3039.4, 3493.1]$	$[\mathbf{299.9}, \mathbf{646.1}]$
F_2	Best	79.74	$\mathbf{3.60}$
	Median	82.39	$\mathbf{4.86}$
	Worse	85.92	$\mathbf{11.91}$
	Mean	82.93	$\mathbf{5.78}$
	Std	$\mathbf{2.09}$	2.37
	95% CI	$[81.59, 84.25]$	$[\mathbf{4.26}, \mathbf{7.28}]$
F_3	Best	$76,615,772.08$	$\mathbf{39,718.90}$
	Median	$119,733,049.20$	$\mathbf{137,279.03}$
	Worse	$169,316,779.50$	$\mathbf{407,661.64}$
	Mean	$123,184,755.70$	$\mathbf{154,306.34}$
	Std	$29,956,737.58$	$\mathbf{114,000.53}$
	95% CI	$[1.06e08, 1.39e08]$	$[\mathbf{0.91e05}, \mathbf{2.17e05}]$
F_4	Best	$5,209.99$	$\mathbf{2,543.51}$
	Median	$5,324.57$	$\mathbf{4,279.56}$
	Worse	$\mathbf{5,388.24}$	$6,003.94$
	Mean	$5,332.59$	$\mathbf{4,216.34}$
	Std	$\mathbf{43.82}$	$1,017.94$
	95% CI	$[5312.1, 5353.1]$	$[\mathbf{3739.9}, \mathbf{4692.7}]$
F_5	Best	24.29	$\mathbf{1.25}$
	Median	24.71	$\mathbf{1.55}$
	Worse	27.59	$\mathbf{2.13}$
	Mean	25.16	$\mathbf{1.75}$
	Std	1.10	$\mathbf{0.37}$
	95% CI	$[24.42, 25.90]$	$[\mathbf{1.49}, \mathbf{1.99}]$
F_6	Best	4.66	$\mathbf{2.49}$
	Median	4.97	$\mathbf{4.12}$
	Worse	$\mathbf{5.15}$	6.73
	Mean	4.94	$\mathbf{4.51}$
	Std	$\mathbf{0.17}$	1.44
	95% CI	$[4.87, 5.00]$	$[\mathbf{3.91}, \mathbf{5.09}]$
F_7	Best	-3683.07	$\mathbf{-3957.85}$
	Median	-3575.13	$\mathbf{-3834.07}$
	Worse	-3565.73	$\mathbf{-3830.36}$
	Mean	-3593.75	$\mathbf{-3851.82}$
	Std	$\mathbf{32.74}$	38.80
	95% CI	$[-3615.7, -3571.8]$	$[\mathbf{-3877.9}, \mathbf{-3825.7}]$



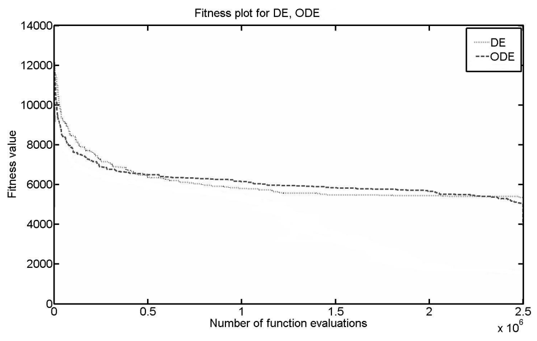
(a) F1



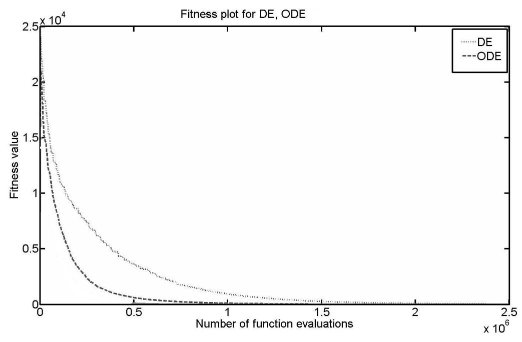
(b) F2



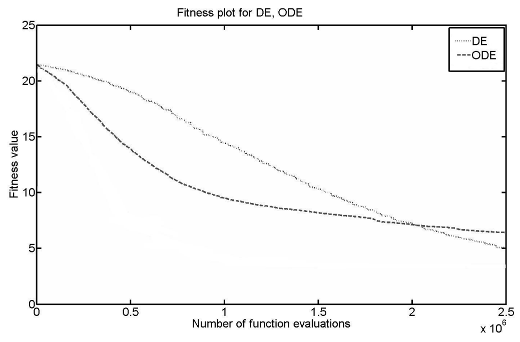
(c) F3



(d) F4



(e) F5

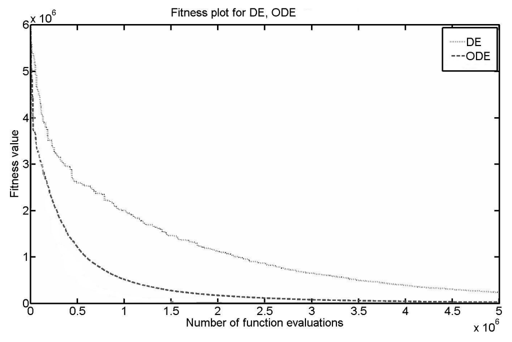


(f) F6

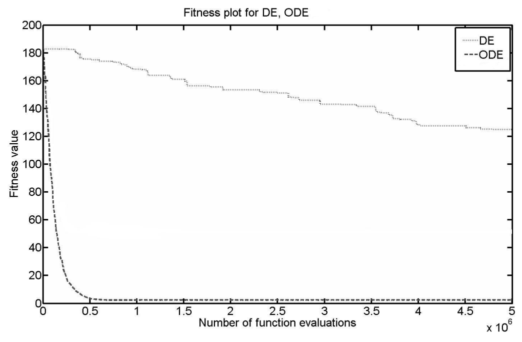
Figure 2: Fitness plots for DE and ODE, 500D.

Table 3: Numerical results for DE and ODE on seven 1000-dimensional minimization problems (25 runs per function). The best result of each error measure is emphasized in boldface. 95% CI stands for 95% confidential interval.

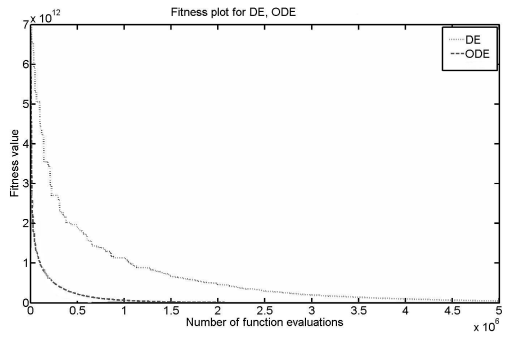
Function	Error Value	DE	ODE
F_1	Best	$223,944.73$	$\mathbf{19,548.90}$
	Median	$236,805.13$	$\mathbf{21,104.67}$
	Worse	$258,806.47$	$\mathbf{43,417.84}$
	Mean	$238,923.73$	$\mathbf{23,903.98}$
	Std	$12,141.16$	$\mathbf{8,009.82}$
	95% CI	$[2.30e05, 2.47e05]$	$[\mathbf{1.81e04}, \mathbf{2.96e04}]$
F_2	Best	119.57	$\mathbf{0.44}$
	Median	121.68	$\mathbf{0.77}$
	Worse	123.11	$\mathbf{2.88}$
	Mean	121.33	$\mathbf{1.31}$
	Std	1.05	$\mathbf{0.94}$
	95% CI	$[120.57, 122.09]$	$[\mathbf{0.64}, \mathbf{1.99}]$
F_3	Best	$35,743,891,601$	$\mathbf{213,105,668}$
	Median	$40,519,312,742$	$\mathbf{572,841,466}$
	Worse	$44,559,917,677$	$\mathbf{1,069,602,053}$
	Mean	$40,215,461,419$	$\mathbf{516,296,792}$
	Std	$2,838,193,442$	$\mathbf{326,088,526}$
	95% CI	$[3.81e10, 4.22e10]$	$[\mathbf{2.83e08}, \mathbf{7.49e08}]$
F_4	Best	$11,589.29$	$\mathbf{5,704.55}$
	Median	$11,791.33$	$\mathbf{5,904.49}$
	Worse	$11,898.50$	$\mathbf{7,309.99}$
	Mean	$11,782.84$	$\mathbf{6,168.10}$
	Std	$\mathbf{105.06}$	620.58
	95% CI	$[1.17e04, 1.18e04]$	$[\mathbf{5.72e03}, \mathbf{6.61e03}]$
F_5	Best	$1,845.36$	$\mathbf{138.99}$
	Median	$2,040.99$	$\mathbf{182.19}$
	Worse	$2,101.89$	$\mathbf{185.05}$
	Mean	$2,016.90$	$\mathbf{179.01}$
	Std	79.15	$\mathbf{14.13}$
	95% CI	$[1.96e03, 2.07e03]$	$[\mathbf{168.90}, \mathbf{189.13}]$
F_6	Best	14.80	$\mathbf{10.07}$
	Median	15.14	$\mathbf{12.64}$
	Worse	15.51	$\mathbf{13.40}$
	Mean	15.13	$\mathbf{12.14}$
	Std	$\mathbf{0.23}$	1.14
	95% CI	$[14.97, 15.30]$	$[\mathbf{11.32}, \mathbf{12.96}]$
F_7	Best	$-6,764.16$	$\mathbf{-7,326.71}$
	Median	$-6,705.17$	$\mathbf{-7,290.84}$
	Worse	$-6,692.63$	$\mathbf{-7,103.89}$
	Mean	$-6,711.71$	$\mathbf{-7,256.45}$
	Std	$\mathbf{21.08}$	87.08
	95% CI	$[-6726, -6696]$	$[\mathbf{-7318}, \mathbf{-7194}]$



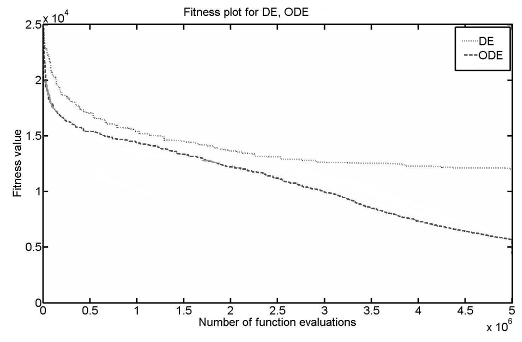
(a) F1



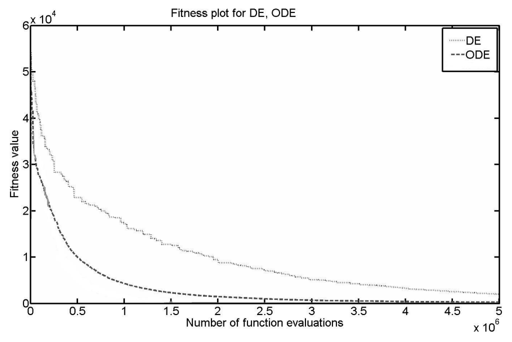
(b) F2



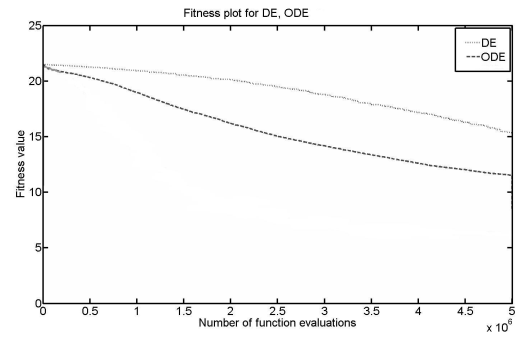
(c) F3



(d) F4



(e) F5



(f) F6

Figure 3: Fitness plots for DE and ODE, 1000D.

Appendix A: List of Bound Constrained Global Optimization High-Dimensional Benchmark Functions [23]

• *Shifted Sphere Function*

$$F_1(X) = \sum_{i=1}^{n} {Z_i}^2 + f\_bias_1,$$
$$X \in [-100, 100], Z = (X - O), X = [x_1, x_2, ..., x_n]$$
$$O = [o_1, o_2, ..., o_n] \text{ : The shifted global optimum.}$$

Global optimum: $X^* = O, F_1(X^*) = f\_bias_1 = -450$
Unimodal, shifted, separable, and scalable.

• *Schwefel's Problem 2.21*

$$F_2(X) = max_i\{|Z_i|, 1 \leq i \leq n\} + f\_bias_2,$$
$$X \in [-100, 100], Z = (X - O), X = [x_1, x_2, ..., x_n]$$
$$O = [o_1, o_2, ..., o_n] \text{ : The shifted global optimum.}$$

Global optimum: $X^* = O, F_2(X^*) = f\_bias_2 = -450$
Unimodal, shifted, non-separable, and scalable.

• *Shifted Rosenbrock's Function*

$$F_3(X) = \sum_{i=1}^{n-1} \{100(Z_i^2 - Z_{i+1})^2 + (Z_i - 1)^2\} + f\_bias_3,$$
$$X \in [-100, 100], Z = (X - O) + 1, X = [x_1, x_2, ..., x_n]$$
$$O = [o_1, o_2, ..., o_n] \text{ : The shifted global optimum.}$$

Global optimum: $X^* = O, F_3(X^*) = f\_bias_3 = 390$
Multi-modal, shifted, non-separable, scalable, and having a very narrow valley from local optimum to global optimum.

• *Shifted Rastrigins Function*

$$F_4(X) = \sum_{i=1}^{n} \{Z_i^2 - 10\cos(2\pi Z_i) + 10\} + f\_bias_4,$$
$$X \in [-5, 5], Z = (X - O), X = [x_1, x_2, ..., x_n]$$
$$O = [o_1, o_2, ..., o_n] \text{ : The shifted global optimum.}$$

Global optimum: $X^* = O, F_4(X^*) = f\_bias_4 = -330$
Multi-modal, shifted, separable, scalable, and local optimas number is huge.

• *Shifted Griewank's Function*

$$F_5(X) = \sum_{i=1}^{n} \frac{Z_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{Z_i}{\sqrt{i}}\right) + 1 + f\_bias_5,$$
$$X \in [-600, 600], Z = (X - O), X = [x_1, x_2, ..., x_n]$$
$$O = [o_1, o_2, ..., o_n] \text{ : The shifted global optimum.}$$

Global optimum: $X^* = O, F_5(X^*) = f\_bias_5 = -180$

Multi-modal, shifted, non-separable, and scalable.

• *Shifted Ackley's Function*

$$F_6(X) = -20 \exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} Z_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=1}^{n}\cos(2\pi Z_i)}{n}\right) + 20 + e,$$

$$X \in [-32, 32], Z = (X - O), X = [x_1, x_2, ..., x_n]$$
$$O = [o_1, o_2, ..., o_n] : \text{The shifted global optimum.}$$

Global optimum: $X^* = O, F_6(X^*) = f\_bias_6 = -140$
Multi-modal, shifted, separable, and scalable.

• *FastFractal "DoubleDip" Function*

$$F_7(X) = \sum_{i=1}^{n} fractal1D(x_i + twist(x_{(\text{i mod n})+1})),$$

$$twist(y) = 4(y^4 - 2y^3 + y^2),$$

$$fractal1D(x) \approx \sum_{k=1}^{3}\sum_{1}^{2^{k-1}}\sum_{1}^{ran2(o)} doubledip\left(x + ran1(o), \frac{1}{2^{k-1}(2 - ran1(o))}\right),$$

$$doubledip(x, c, s) = \begin{cases} (-6144(x-c)^6 + 3088(x-c)^4 - 392(x-c)^2 + 1) \times s & \text{if } -0.5 < x < 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

$X = [x_1, x_2, ..., x_n]$,
o: integer, seeds the random generators
$ran1(o)$: double, pseudorandomly chosen, with seed o, with equal probability from the interval $[0, 1]$.
$ran2(o)$: integer, pseudorandomly chosen, with seed o, with equal probability from the set $\{0, 1, 2\}$.
$fractal1D(x)$ is an approximation to a recursive algorithm, it does not take account of wrapping at the boundaries, or local re-seeding of the random generators.
X^* =unknown, $F_7(X^*)$ = unknown.
Multi-modal, non-separable, and scalable.