Opposition Based Chaotic Differential Evolution Algorithm for Solving Global Optimization Problems

[1]Radha Thangaraj, [1]Millie Pant, [1]Thanga Raj Chelliah and [2,3]Ajith Abraham
[1]Indian Institute of Technology Roorkee, Roorkee – 247 667, India
[2]Machine Intelligent Research Labs (MIR Labs), Scientific Network for Innovation and Research Excellence, USA
[3]IT For Innovations, VSB - Technical University of Ostrava, Czech Republic
E-mail: t.radha@ieee.org, millifpt@iitr.ernet.in, thangaraj@ieee.org, ajith.abraham@ieee.org

Abstract— **A modified differential evolution (DE) algorithm based on opposition based learning and chaotic sequence named Opposition based Chaotic Differential Evolution (OCDE) is proposed. The proposed OCDE algorithm is different from basic DE in two aspects. First is the generation of initial population, which follows Opposition Based Learning (OBL) rules; and the second is: dynamic adaption of scaling factor F using chaotic sequence. The numerical results obtained by OCDE when compared with the results obtained by DE and ODE (opposition based DE) algorithms on eighteen benchmark function demonstrate that the OCDE is able to find a better solution while maintaining a reasonable convergence rate.**

Keywords-differential evolution, chaotic sequence, opposition based learning, global optimization

I. INTRODUCTION

Evolutionary algorithms (EAs) are stochastic search methods that mimic evolutionary processes encountered in nature. The common conceptual base of these methods is to evolve a population of candidate solutions by simulating the processes involved in the evolution of genetic material of organisms', such as natural selection and biological evolution. EAs can be characterized as global optimization algorithms because of their ability to obtain the global optimal. Their population-based nature allows them to find global optimal solutions with more probability in comparison to classical gradient based methods. EAs have been successfully applied to a wide range of optimization problems, such as image processing, pattern recognition, scheduling, and engineering design [1], [2]. The most prominent EAs proposed in the literature are genetic algorithms [1], evolutionary programming [3], evolution strategies [4], genetic programming [5] and differential evolution [6], [7].

In this research, we have concentrated our work on DE, introduced by Storn and Price for the real parameter optimization [7]. DE is a novel evolutionary approach capable of handling non-differentiable, non-linear and multimodal objective functions. DE has been designed as a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of EAs. It typically requires few, easily chosen control parameters. Experimental results have shown that performance of DE is better than many other well-known EAs [8], [9]. While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process [10]. With the advantage of faster convergence speed, less adjustable parameters, better robustness and simpler algorithm, the DE algorithm has been achieved fine application effects in neural network training, filter design, cluster analysis etc.

However, it has been observed that despite having several attractive features, DE sometimes does not perform as good as the expectations. Empirical analysis of DE has shown that it may stop proceeding towards a global optimum though the population has not converged even to a local optimum [11]. The situation when the algorithm does not show any improvement though it accepts new individuals in the population is known as *stagnation*. Besides this, DE also suffers from the problem of premature convergence. This situation arises when there is a loss of diversity in the population. It generally takes place when the objective function is multi modal having several local and global optima. Further, like other EA, the performance of DE deteriorates with the increase in dimensionality of the objective function. Consequently, several modifications have been made in the structure of DE to improve its performance [12 - 15].

In continuation to the techniques of improving the performance of DE, in this paper, we present a modified version of DE called Opposition based Chaotic Differential Evolution (OCDE) algorithm.

The proposed OCDE algorithm is different from basic DE in two aspects. Firstly, in OCDE, the population of individuals is initialized using Opposition Based Learning (OBL) rule; and secondly, it dynamically adapts the scale factor F using chaotic sequence.

The structure of the paper is as follows: in Section II, we briefly explain the Differential Evolution Algorithm, in Section III; we have defined and explained the proposed OCDE algorithm. Section IV deals with experimental settings, Section V gives the benchmark problems and their numerical results and finally the paper concludes with Section VI.

II. DIFFERENTIAL EVOLUTION ALGORITHM

DE shares a common terminology of selection, crossover and mutation operators with GA however it is the application of these operators that make DE different from GA; while, in GA crossover plays a significant role, it is the mutation operator which affects the working of DE [16]. Also DE has a unique selection strategy that makes it different from GA and other EAs.

The working of basic DE may be described as follows:

For a D-dimensional search space, each target vector $x_{i,g}$, a mutant vector is generated as:

$$V_{i,g} = X_{r_1,g} + F * (X_{r_2,g} - X_{r_3,g}) \quad (1)$$

where $r_1, r_2, r_3 \in \{1,2,....,NP\}$ are randomly chosen integers, must be different from each other and also different from the running index i.

F (>0) is a scaling factor which controls the amplification of the differential evolution $(x_{r_2,g} - x_{r_3,g})$. In order to increase the diversity of the perturbed parameter vectors, crossover is introduced. The parent vector is mixed with the mutated vector to produce a trial vector $u_{ji,g+1}$,

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if} \quad rand_j \le Cr \vee j = k \\ x_{j,i,G} & \text{otherwise} \end{cases} \quad (2)$$

where j = 1, 2,……, D; $rand_j \in [0,1]$; CR is the crossover constant takes values in the range [0, 1] and $j_{rand} \in (1,2,....., D)$ is the randomly chosen index.

Selection is the step to choose the vector by holding a tournament between the target vector and the corresponding trial vector with the aim of creating an individual for the next generation. The one with better function value survives the next generation. Thus, the population in the next generation is better or is at least as good as the previous generation.

III. OPPOSITION BASED CHAOTIC DIFFERENTIAL EVOLUTION ALGORITHM

Before explaining the proposed OCDE algorithm in detail, we explain the concept of OBL and chaos which are used in OCDE algorithms.

A. *Opposition based learning*

The concept of opposition-based learning (OBL) was introduced by Tizhoosh [17] and has thus far been applied to accelerate reinforcement learning, backpropagation learning and DE [12]. The main idea behind OBL is the simultaneous consideration of an estimate and its corresponding opposite estimate in order to to achieve a better approximation of the current candidate solution.

If $X = (x_1, x_2,....x_n)$ is a point in *n*-dimensional space, where $x_1, x_2,....x_n \in R$ *and* $x_i \in [a_i, b_i]$, *i* = 1‹2‹ ≤ ≤ ≤ ‹ *n*,

Then the opposite point $\breve{X} = (\breve{x}_1, \breve{x}_2,....\breve{x}_n)$ is defined by its components as $\breve{X} = a_i + b_i - x_i$.

Now, by employing the opposite point definition, the opposition-based learning can be used in optimization as:

- Generate a point $X = (x_1, x_2,....x_n)$ and its opposite point $\breve{X} = (\breve{x}_1, \breve{x}_2,....\breve{x}_n)$ in an n-dimensional search space.
- Evaluate the fitness of both points *f(X)*and $f(\breve{X})$.
- If $f(X) \le f(\breve{X})$, then replace X with $\breve{X}$; otherwise , continue with *X.*

Thus, the point and its opposite point are evaluated simultaneously in order to continue with the fitter one.

B. *Chaotic sequence*

Chaos is a phenomenon that can appear in solutions for nonlinear differential equations. An essential feature of chaotic systems is that small changes in the parameters or the starting values for the data lead to vastly different future behaviors, such as stable fixed points, periodic oscillations, bifurcations, and ergodicity. These behaviors can be analyzed based on Lyapunov exponents and the attractor theory.

Logistic mapping, which is used to generate chaotic sequence is the most commonly used generator [18] is defined as follows:

$$z(t+1) = \eta * z(t) * [1 - z(t)] \quad (3)$$

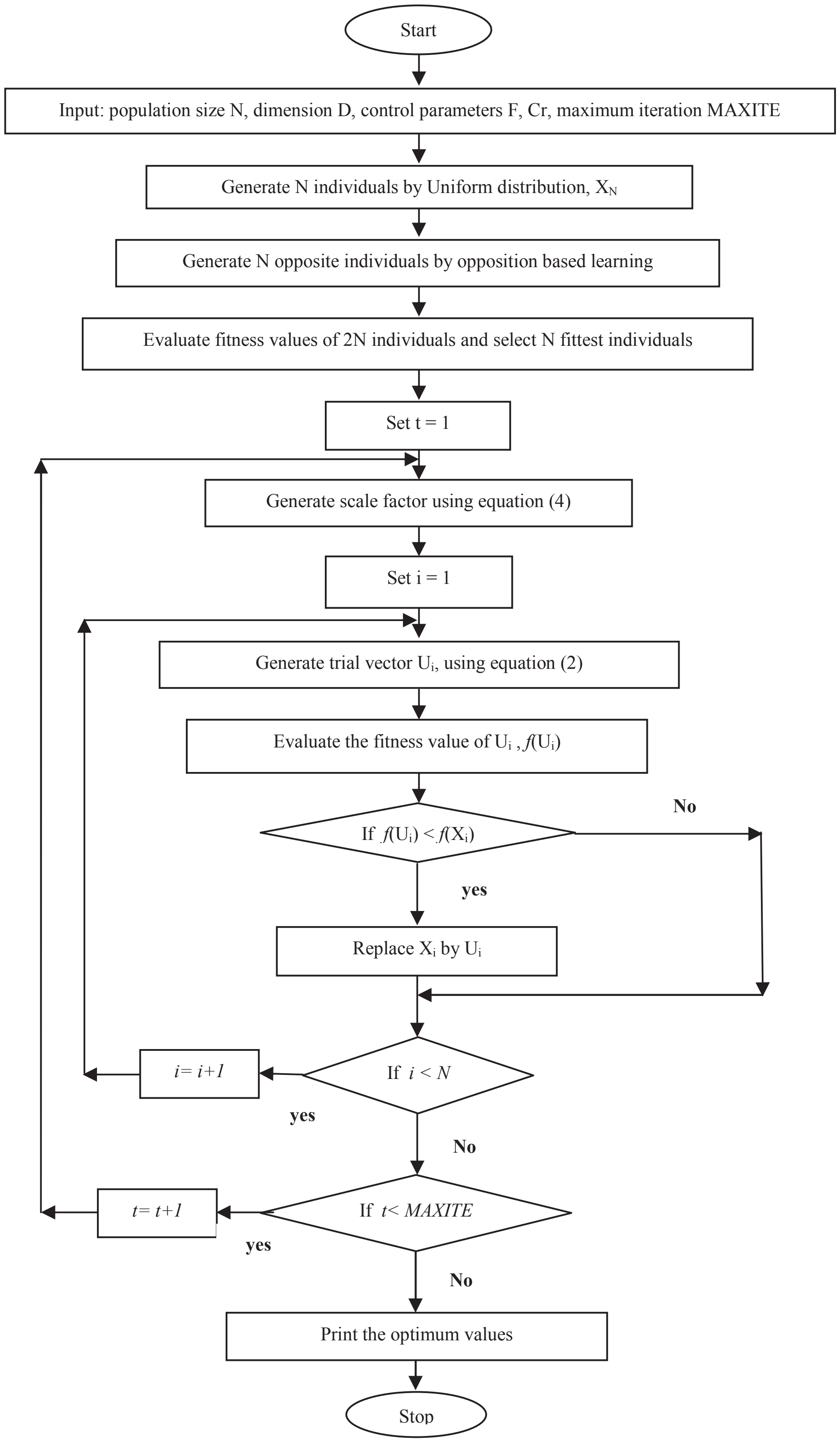
where η is the chaos attractor, and the chaos space is [0,1]. The behavior of the system of (3) is greatly changed with the variation of η. The value of η determines whether z stabilizes at a constant size, oscillates between a limited sequence of sizes, or behaves chaotically in an unpredictable pattern. A very small difference in the initial value of *z* causes substantial differences in its long-time behavior [19]. Equation (3) is deterministic, displaying chaotic dynamics when η = 4 and $z(1) \notin \{0, 0.25, 0.50, 0.75, 1\}$. In this case, $z(t)$ is distributed in the range (0,1) provided the initial $z(1) \in (0,1)$.

C. *OCDE algorithms*

The OCDE algorithm is the simple extension of basic DE algorithms. The basic operators of OCDE algorithms are same as of basic DE except for two points which are:

(1) Population initialization: According to the literature review about population initialization [12], random number generation, in absence of an *a priori* knowledge, is a widely used choice to create an initial population. However, the computation time and the convergence speed are affected by the choice of initial individuals. Therefore, as mentioned in section III.A, by utilizing OBL we can obtain fitter starting candidate solutions even when there is no knowledge about the solution(s).

In OCDE, the initial population is generated using OBL, which is in contrast to basic DE where the population is generated using uniformly distributed random numbers.



Start
Input: population size N, dimension D, control parameters F, Cr, maximum iteration MAXITE
Generate N individuals by Uniform distribution, X_N
Generate N opposite individuals by opposition based learning
Evaluate fitness values of 2N individuals and select N fittest individuals
Set t = 1
Generate scale factor using equation (4)
Set i = 1
Generate trial vector U_i, using equation (2)
Evaluate the fitness value of U_i , $f(U_i)$
If $f(U_i) < f(X_i)$
No
yes
Replace X_i by U_i
If $i < N$
$i= i+1$
yes
No
If $t< MAXITE$
$t= t+1$
yes
No
Print the optimum values
Stop

Figure 1. Flowchart of OCDE algorithm

In OCDE, a population of 2N individuals is generated using uniform random distribution and their opposite individuals, out of which N best individuals are selected.

(2) Chaos scale factor: The scaling factor F plays a significant role in the generation of perturbed mutant vector. Its role is to guide the individuals towards the optimum. A careful choice of F, helps in maintaining a balance between exploration and exploitation. In the initial studies, F was taken as a constant. As the researchers observed the crucial role of F, dynamic and adaptive scaling factors were also proposed [13]-[15], [20] etc. In OCDE, F modified as follows:

$$F(t+1) = \eta * F(t) * [1 - F(t)] \text{, where } \eta = 4. \quad (4)$$

The use of chaos makes the scaling factor adaptive and more random in nature, which helps in better exploration and exploitation of the search space.

A combination of OBL and chaotic scaling factor is expected to find better fitness values while maintaining the diversity. The flowchart of OCDE algorithm is given in Figure 1. The sample points generated by using chaos sequence for different chaos attractor are given in Figure 2.

IV. EXPERIMENTAL SETTINGS

The parameter settings of DE and OCDE algorithms are given in Table I. The benchmark of eighteen problems is given in Table II with solution space and true optimum value.

TABLE I. EXPERIMENTAL SETTINGS

Item	OCDE	DE settings
Initialization	OBL	Uniform distribution
Population size	100	100
Scale factor F	Chaotic sequence	0.5
Crossover rate Cr	0.9	0.9
Maximum number of function evaluations	10^6	10^6
Value to reach (VTR)	10^{-8}	10^{-8}
Total runs	25	25

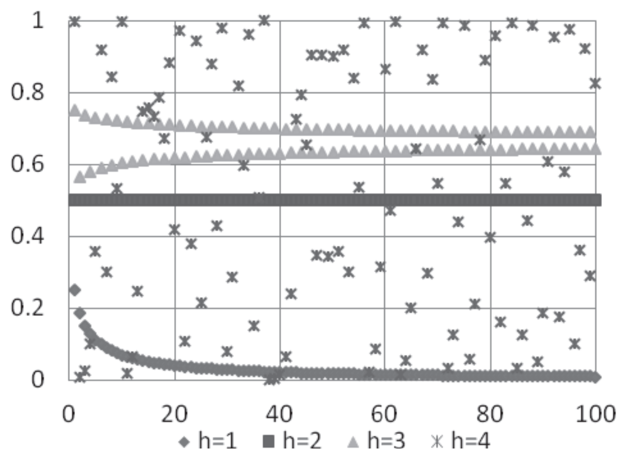


Figure 2. Sample points generated using Chaos sequence for different chaos attractor η

TABLE II. TEST SUIT OF BENCHMARK PROBLEMS

Function	Solution space	True optima
$f_1(x) = \sum_{i=1}^{n} x_i^2$	[-5.12, 5.12]	0
$f_2(x) = \sum_{i=1}^{n} i x_i^2$	[-5.12, 5.12]	0
$f_3(x) = \sum_{i=0}^{n-1} (\sum_{j=0}^{i} x_j)^2$	[-100, 100]	0
$f_5(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12, 5.12]	0
$f_6(x) = \frac{1}{4000} \sum_{i=0}^{n-1} x_i^2 - \sum_{i=0}^{n-1} \cos(\frac{x_i}{\sqrt{i+1}}) + 1$	[-600, 600]	0
$f_7(x) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$	[-32, 32]	0
$f_8(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5 i x_i)^2 + (\sum_{i=1}^{n} 0.5 i x_i)^4$	[-10, 10]	0
$f_9(x) = \sum_{i=0}^{n-1} \lvert x_i \rvert + \prod_{i=0}^{n-1} \lvert x_i \rvert$	[-10, 10]	0
$f_{10}(x) = \sum_{i=0}^{n-1} \lfloor x_i + 1/2 \rfloor^2$	[-100, 100]	0
$f_{11}(x) = \sum_{i=1}^{n} \lvert x_i \sin(x_i) + 0.1 x_i \rvert$	[-10, 10]	0

TABLE II. TEST SUIT OF BENCHMARK PROBLEMS (CONTINUED)

Function	Solution space	True optima
$f_{12,13}(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i\frac{x_i^2}{\pi}))^{2m}$, m=0	[-π, π]	-9.66015 if n = 10 -1.8013 if n = 2
$f_{14}(x) = \{1+(x_0+x_1+1)^2(19-14x_0+3x_0^2-14x_1+6x_0x_1+3x_1^2)\}$ $\{30+(2x_0-3x_1)^2(18-32x_0+12x_0^2+48x_1-36x_0x_1+27x_1^2)\}$	[-2, 2]	3
$f_{15}(x) = (x_1 - \frac{5.1}{4\pi^2}x_0^2 + \frac{5}{\pi}x_0 - 6)^2 + 10(1-\frac{1}{8\pi})\cos(x_0) + 10$	[-10, 10]	0.39788
$f_{16}(x) = \sum_{j=1}^{5} j\cos((j+1)x_1 + j)\sum_{j=1}^{5} j\cos((j+1)x_2 + j)$	[-10, 10]	-186.7309
$f_{17}(x) = -\sum_{i=1}^{n}\sum_{j=1}^{5} j\sin((j+1)x_i + j)$	[-10, 10]	-24.06249
$f_{18}(x) = -\sum_{i=1}^{4}\alpha_i \exp(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2)$	[0, 1]	-3.86278
For f_{18}, $\alpha = [1 \quad 1.2 \quad 3 \quad 3.2]$, $A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$, $P = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$		

V. RESULTS ANALYSIS

In order to compare the proposed OCDE schemes with basic DE we considered various performance metrics like average fitness function value and standard deviation (STD) to check the efficiency and reliability of the algorithm. To compare the convergence speed of algorithms we considered the average number of function evaluations (NFE) and the acceleration rate (AR_{NFE}). The speed of the algorithm is also measured by recording the total CPU time. Besides this we have measured the success rate (SR). Where, a run is considered to be a success if the algorithm successfully reaches the value VTR. The definition of the performance measure AR_{NFE} used in Table III and IV is given below:

$$\text{Acceleration rate } (AR_{NFE}) = \frac{\text{Total NFE for basic DE / ODE}}{\text{Total NFE for OCDE}}$$

Table III lists the experimental results in terms of the performance metrics mentioned above for the test bed given in Table I for DE and OCDE.

From the results of Table III, it can be seen that the convergence precision (reflected in terms of Fitness) of OCDE is significantly superior to DE for all the test problems especially for f_4. Compared with DE, the mean fitness values of OCDE are smallest on all the test functions. In case of f_4, there is an improvement of 99.99% in the function value while using the OCDE algorithm. Thus, the global search ability of OCDE outperforms the basic DE. We can also see that the stability (reflected in terms of STD) of OCDE is better than DE. OCDE has smallest STD on all the test functions. Thus, we can conclude that OCDE is more stable and thus more robust than the DE algorithm. Furthermore, we can see that the convergence speed (reflected in terms of NFE and Time) of OCDE algorithm is much better than that of DE algorithm.

In Table IV, the OCDE algorithm is compared with ODE (opposition based DE) [12] in terms of NFE, SR and AR_{NFE}. From this table also, it can be seen that the superior performance of the OCDE algorithm. The concept of acceleration rate (AR_{NFE}) is a criterion to measuring the convergence speed of an algorithm. When AR_{NFE} is greater than 1, it means that the proposed algorithm is better than the basic algorithm. For OCDE algorithm (in comparison with DE and ODE) the AR_{NFE} is greater than 1 on all the test functions except for the test problem f_1, f_2 and f_6 in comparison with ODE.

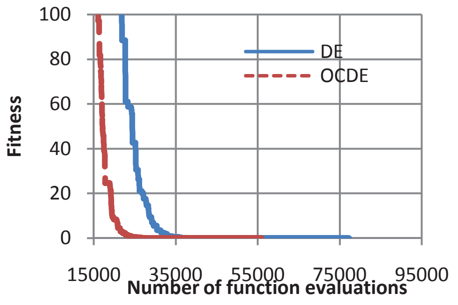


Figure 2 (a) Function f_3

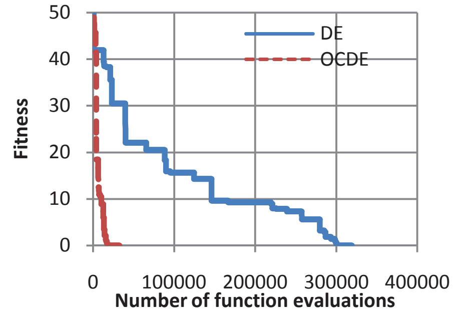


Figure 2 (b) Function f_4

Figure 3. Convergence graphs of selected benchmarks

TABLE III. NUMERICAL RESULTS OF BENCHMARK PROBLEMS $F_1 – F_{18}$

Function	Dimension	DE					OCDE					AR_{NFE}
		Fitness	STD	NFE	SR	Time	Fitness	STD	NFE	SR	Time	
f_1	30	8.8426e-9	1.0930e-9	83070	1	4.2	8.8188e-9	9.7838e-10	52520	1	2.9	1.581683
f_2	30	9.2126e-9	7.3863e-9	92100	1	4.8	8.7090e-9	1.0520e-9	58170	1	3	1.58329
f_3	20	9.2530e-9	8.3044e-10	77030	1	2.8	8.1805e-9	4.5070e-10	52060	1	1.9	1.479639
f_4	10	0.09949	0.29848	426740	0.9	8.1	7.7288e-9	1.3048e-9	32200	1	0.6	13.2528
f_5	30	9.2782e-9	4.7246e-10	108150	1	6.1	8.5514e-9	9.4286e-10	67060	1	3.8	1.612735
f_6	30	6.6726e-9	2.6015e-9	20010	1	1.0	5.8232e-9	1.4864e-9	14670	1	0.7	1.364008
f_7	30	9.3999e-9	4.7545e-10	161520	1	9.1	9.4611e-9	4.0691e-10	95900	1	5.6	1.684254
f_8	30	8.5206e-9	1.2295e-9	363990	1	2.3	9.7039e-9	2.5232e-10	269830	1	1.9	1.34896
f_9	30	8.9373e-9	6.7219e-10	174470	1	0.7	8.8898e-9	6.4392e-10	82740	1	0.3	2.108654
f_{10}	30	0.0000	0.0000	38080	1	0.5	0.0000	0.0000	22670	1	0.3	1.679753
f_{11}	30	9.4923e-9	1.1312e-9	107840	1	0.7	9.0053e-9	4.2872e-10	62780	1	0.5	1.717745
f_{12}	10	-9.6539	0.01366	515200	0.6	13.9	-9.6601	1.4487e-4	124490	0.9	3.2	4.138485
f_{13}	2	-1.80125	2.6769e-5	2560	1	0.03	-1.80125	3.2041e-5	2020	1	0.01	1.267327
f_{14}	2	3.0	2.1118e-9	4290	1	0.02	3.0	2.6630e-9	3940	1	0.01	1.088832
f_{15}	2	0.3979	3.3205e-5	3040	1	0.1	0.3979	2.2903e-5	2910	1	0.1	1.044674
f_{16}	2	-24.0624	3.8123e-5	9690	1	0.03	-24.0625	3.2131e-5	6900	1	0.01	1.404348
f_{17}	2	-186.731	1.8702e-4	16360	1	0.11	-186.731	2.6519e-5	14230	1	0.1	1.149684
f_{18}	3	-3.86273	2.9080e-5	2310	1	0.1	-3.86273	2.4742e-5	2160	1	0.01	1.069444

TABLE IV. COMPARISON RESULTS OF OCDE VS ODE

Function	OCDE		ODE [12]		AR_{NFE}
	NFE	SR	NFE	SR	
f_1	52520	1	47716	1	0.9085301
f_2	58170	1	53304	1	0.9163486
f_3	52060	1	168680	1	3.2401076
f_4	32200	1	70389	0.76	2.1859938
f_5	67060	1	69342	0.96	1.0340292
f_6	14670	1	8328	1	0.5676892
f_7	95900	1	98296	1	1.0249844
f_8	269830	1	369104	1	1.3679131
f_9	82740	1	155636	1	1.8810249
f_{10}	22670	1	23124	1	1.0200265
f_{11}	62780	1	337532	1	5.3764256
f_{12}	124490	0.9	213330	0.56	1.7136316

VI. CONCLUSION

In this paper, a opposition based chaotic differential evolution algorithm is presented, which uses the concept of OBL for initializing the population and uses chaotic sequence to for the scaling factor F. The performance of the proposed OCDE algorithm is validated on a test suit of eighteen benchmark problems taken from literature. Numerical results show that the proposed approach improves the global search ability, stability as well as the convergence speed of the basic DE and maintains a quality of solution. The efficiency of the OCDE algorithm is further compared with the ODE algorithm. In this comparison also, the OCDE algorithm gave a better performance in comparison to ODE.

REFERENCES

[1] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[2] T. B¨ack, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Oxford, U.K.: Oxford University Press, 1997.

[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[4] H.-G. Beyer and H.-P. Schwefel, “Evolution strategies: A comprehensive introduction,” *Natural Comput.*, vol. 1, no. 1, pp. 3–52, May 2002.

[5] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[6] R. Storn and K. Price, “Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces,” *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[7] R. Storn and K. Price, “Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces,” Int. Comput. Sci. Instit., Berkeley, CA, Tech. Rep. TR-95-012, 1995.

[8] R. Storn and K. Price, Differential Evolution – a simple and efficient Heuristic for global optimization over continuous spaces. *Journal Global Optimization.* 11:341 – 359 (1997).

[9] R. Stom, System design by constraint adaptation and differential evolution. *IEEE Transactions on Evolutionary Computation*, 3:22-34 (1999).

[10] A.P. Engelbrecht, *Fundamentals of Computational Swarm Intelligence.* John Wiley & Sons Ltd, 2005.

[11] J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm, Proceedings of MENDEL 2000, 6th

International Mendel Conference on Soft Computing, Brno, Czech Republic Pavel Ošmera, (ed.), pp. 76 – 83, June 7–9, 2000.

[12] S. Rahnamayan, H.R. Tizhoosh, and M. M. A. Salama, Opposition-Based Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 12:64 – 79 (2008).

[13] M. Omran, A. Salman, and A. P. Engelbrecht, Self-adaptive differential evolution, computational intelligence and security, PT 1, Proceedings Lecture Notes In Artificial Intelligence 3801: 192-199, 2005.

[14] J. Brest, S. Greiner, B. Boškovic, M. Mernik, and V. Žumer, Self-adapting Control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10:646 – 657 (2006).

[15] R. Thangaraj, M. Pant and A. Abraham, New Mutation Schemes for Differential Evolution Algorithm and their application to the Optimization of Directional Overcurrent Relay Settings", Applied Mathematics and Computation, Elsevier Science, Vol. 216 (2), pp. 532 – 544, 2010.

[16] D. Karaboga and S. Okdem, A simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm. *Turk J. Elec. Engin.* 12:53 – 60 (2004).

[17] H. R. Tizhoosh, "Opposition-based learning,: A new scheme for machine intelligence", in *Proc. Computational Intelligence for Modeling Control and Automation Conference, Vienna, Austria*, vol. I, pp. 695-701, 2005.

[18] Lds. Coelho and V. C. Mariani, "Combining of chaotic differential evolution and quadratic programming for economic dispatch optimization with valve-point effect", *IEEE Trans. on Power Systems*, Vol. 21 (2), pp. 989 – 996, 2006.

[19] X. F. Yan, D. Z. Chen, and S. X. Hu, ''Chaos-genetic algorithm for optimizing the operating conditions based on RBF-PLS model,'' *Comput. Chem. Eng.*, vol. 27, pp. 1393–1404, Oct. 2003.

[20] S. Das, A. Abraham, U. Chakraborty and A. Konar, Differential Evolution Using a Neighborhood based Mutation Operator, IEEE Transactions on Evolutionary Computation, IEEE Press, USA, Volume 13, Issue 3, pp. 526-553, 2009.