INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 7, ISSUE 3 , MARCH 2018 ISSN 2277-8616

An Alternative Differential Evolution Algorithm (Ade)

Hegazy Zaher, Nissrine Barrak

Abstract: this paper proposes an alternative differential evolution algorithm for solving unconstrained optimization problems. The performance of the given algorithm is measured by the result of 15 benchmarking problems the obtained results are competent in both accuracy and CPU time. The results obtained using the proposed algorithm are more accurate and use less number of function's evaluations compared with several algorithms.

Keywords: benchmarks, differential evolution, crossover rate, evolutionary algorithm global optimization, population size, scale factor.

1 Introduction

TNE of the most fundamental principles in our world is the search for an optimal state . Optimization can be thought of as the process of attempting to find the best possible solution amongst all those available. Therefore, the task of optimization is to model the problem in terms of some evaluation function and then employ a search algorithm to minimize (or maximize, depending on the problem) that objective function. Global optimization algorithms are methods to find optimal solutions for giving problems. Evolutionary Algorithm can deal with complex optimization problems better than traditional optimization techniques. The key aspect distinguishing an evolutionary search algorithm from such traditional algorithms is that it is population-based. Differential evolution (DE) is an evolutionary algorithm proposed by Storn and Price (1997). Differential Evolution is as a stochastic direct search method using population or multiple search points. It has been successfully applied to the optimization problems, including nonlinear, non-differentiable and non-convex by Chakraborty, (2008). Differential evolution is a small and simple mathematical model of a big and naturally complex process of evolution. Differential Evolution seems particularly appropriate to solve unconstrained optimization problems; one of the main features of differential evolution is its low number of parameters: the population size, the maximum number of generations, the recombination rate, and the mutation rate. Therefore, the differential evolution is a simple and fast algorithm for unconstrained optimization problems and, in order to extend its domain of application to practical problems in various fields. Differential Evolution is simple and efficient algorithm especially for global optimization over continuous search spaces, although the standard differential evolution is reliable algorithm but it may enhanced to make new algorithm better than the standard in run time and more than accurate, differential evolution influence by its control parameter population size, scale factor, and crossover rate for this reason the paper used scale factor F, the crossover rate CR to enhance Differential Evolution.

- *Professor of Mathematical Statistics, Institute of Statistical Studies and Research, Cairo University.*
- *M.SC. Researcher, Institute of Statistical Studies and Research, Cairo University.*

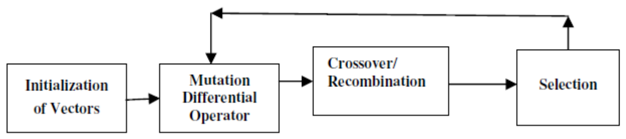
2 Literature Review

The first paper about genetic annealing was published in the October 1994 issue of Dr. Dobb's Journal. It was a population-based combinatorial algorithm that realized an annealing criterion via thresholds driven by the average performance of the population. For the first time differential evolution was described by Price and Storn in the International Computer Science Institute technical report is differential evolution — A simple and efficient adaptive scheme for global optimization over continuous spaces (1995). The success of Differential evolution was demonstrated at the First International Contest on Evolutionary Optimization in May of (1996), which was held in conjunction with the International Conference on Evolutionary Computation (1996). The algorithm won third place for proposed benchmarks. Also, their article for the Journal of Global Optimization Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces was soon published, in December (1997), these papers introduced Differential evolution to a large international public and demonstrated the advantages of Differential evolution over the other heuristics. In (1999), I. Zelinka and J. Lampinen described a simple and, at the same time, efficient way of handling simultaneously continuous, integer, and discrete variables. The ideas of "directions" were spontaneously grasped by H.Y. Fan and J. Lampinen. They proposed alternations of the classical strategy (the first strategy suggested by Price) with a triangle mutation scheme in (2001). The next important point is control parameters; Differential evolution disposes three control parameters: population size, differentiation constant, and crossover constant, by J.Liu, J. Lampinen "On setting the control parameter of the differential evolution method", (2002). A new version of Differential Evolution which eliminates the need for manual parameter tuning is proposed. J. TEO and M. Yunus Hamid present a first attempt at self-adapting the population size parameter in addition to self-adapting crossover and mutation rates (2005). Josef Tvrd´ık published paper about the adaptation of control parameters in differential evolution (2007). The competition of different control parameter settings was proposed in order to ensure the self-adaptation of parameters within the search process. A new version of Differential Evolution which eliminates the need for manual parameter tuning is proposed. J. TEO and M. Yunus Hamid present a first attempt at self-adapting the population size parameter in addition to self-adapting crossover and mutation rates (2005). Josef Tvrd´ık published paper about the adaptation of control parameters in differential evolution (2007). The competition of different control parameter settings was proposed in order to ensure the self-adaptation of parameters within the search process. The performance of the proposed approach is investigated and compared with other well-known approaches. The results show that the new algorithm provides good performance when applied to multimodal problems with the added advantage that no parameter tuning is needed. Shahryar Rahnamayan, Hamid R. Tizhoosh, Magdy M.A. Salama present a new algorithm to improve the efficiency of Differential Evolution to cope with noisy optimization problems. It employs opposition-based learning for population initialization, generation jumping, and also improving population's best member (2006). Zhenyu Yang, Ke Tang and Xin Yao propose two new efficient Differential Evolution variants, for high-dimensional optimization (up to 1000 dimensions) in (2007). In the paper by Adam Slowik and Michal Bialko an application of differential evolution algorithm to training of artificial neural networks is presented (2008). A. Cichoń, E. Szlachcic, I. F. Kotowski present paper for Differential evolution for multi-objective optimization with self-adaptation, the paper an adapted version of the differential evolution algorithm has been created to solve a multi-objective optimization problem (2010). T. Warren Liao presents two hybrid differential evolution algorithms for optimizing engineering design problems. One hybrid algorithm enhances a basic differential evolution algorithm with a local search operator, while the other adding a second metaheuristic to cooperate with the differential evolution algorithm so as to produce the desirable synergetic effect. Musrrat Ali, Millie Pant and Ajith Abraham propose a modified variant of Differential Evolution algorithm called improved differential evolution (IDE). It works in three phases: decentralization, evolution and centralization of the population. Initially, the individuals of the population are partitioned into several groups of subpopulations (decentralization phase) through a process of shuffling. Each subpopulation is allowed to evolve independently from each other with the help of Differential Evolution (evolution phase). Periodically, the subpopulations are merged together (centralization phase) and again new subpopulations are reassigned to different groups. These three phases help in searching all the potential regions of the search domain effectively, thereby, maintaining the diversity (2011).

3 The Standard Differential Evolution (DE)

DE is a simple evolutionary algorithm. It works through a simple cycle of stages, presented in Fig (1)

Fig.1



3.1 Initialization of the Parameter Vectors

DE searches for a global optimum point in a D-dimensional continuous hyperspace. It begins with a randomly initiated population of NP D dimensional real-valued parameter vectors. Each vector, also known as genome/chromosome, forms a candidate solution to the multidimensional optimization problem. subsequent generations in DE represented by discrete time steps like t = 0, 1,2 ...t, t+1 etc (in most of the DE literatures, the successive generations are represented by G, G+1, G+2…or g, g+1 etc. [13] but we adopt a slightly different notation in order to remain consistent with the notations used in other chapters of the Volume and also to facilitate the mathematical analysis of DE undertaken here. Since the parameter vectors are likely to be changed over different generations, we adopt the following notation for representing the i-th vector of the population at the current generation (i.e. at time t = t) as:

$$\bar{X}_i(t) = [x_{1,1}(t), x_{i,2}(t), \ldots, x_{i,D}(t)]^T, \qquad (1)$$

[illegible]

For each parameter of the problem, there may be a certain range within which value of the parameter should lie for better search results. At the very beginning of a DE run or at t=0, problem parameters or independent variables are initialized somewhere in their feasible numerical range. So, if the j-th parameter of the given problem has its lower and upper bound as $x_{min,\ j}$, and $x_{max,j}$ respectively and rand $_{i,\ j}$ (0,1) denotes the j-th instantiation of a uniformly distributed random number lying between 0 and 1 for the i-th vector, then we may initialize the j-th component of the i-th population members as:

$$x_{i,j}(0) = x_{\min,j} + rand_{i,j}(0,1).(x_{\max,j} - x_{\min,j}) \qquad (2)$$

3.1 Initialization of the Parameter Vectors

DE searches for a global optimum point in a D-dimensional continuous 0hyperspace. It begins with a randomly initiated population of NP D dimensional real-valued parameter vectors. Each vector, also known as genome/chromosome, forms a candidate solution to the multidimensional optimization problem. subsequent generations in DE represented by discrete time steps like t = 0, 1,2 ...t, t+1 etc (in most of the DE literatures, the successive generations are represented by G, G+1, G+2…or g, g+1 etc. [13] but we adopt a slightly different notation in order to remain consistent with the notations used in other chapters of the Volume and also to facilitate the mathematical analysis of DE undertaken here. Since the parameter vectors are likely to be changed over different generations, we adopt the following notation for representing the i-th vector of the population at the current generation (i.e. at time t = t) as:

$$\bar{X}_i(t) = [x_{1,1}(t), x_{i,2}(t), \ldots, x_{i,D}(t)]^T, \qquad (3)$$

with i = 1, 2,…, NP. For each parameter of the problem, there may be a certain range within which value of the parameter should lie for better search results. At the very beginning of a DE run or at t = 0, problem parameters or independent variables are initialized somewhere in their feasible numerical range. So, if the j-th parameter of the given problem has its lower and upper bound as xmin, j, and xmax,j respectively and rand i, j (0,1) denotes the j-th instantiation of a uniformly distributed random number lying between 0 and 1 for the i-th vector, then we may initialize the j-th component of the i-th population members as,

$$x_{i,j}(0) = x_{\min,j} + rand_{i,j}(0,1).(x_{\max,j} - x_{\min,j}) \qquad (4)$$

Mutation for each population vector xi(G) creates a mutant vector vi(G)

$$\mathbf{v}_i^{(G)} = \{v_{i,1}^{(G)}, v_{i,2}^{(G)}, \ldots, v_{i,D}^{(G)}\},\ i = 1,2,\ldots,NP.$$

A new mutant vector can be created using one of the mutation strategies.

(6

$$DE/rand/1: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F\cdot(\mathbf{x}_{r_2}^{(G)} - \mathbf{x}_{r_3}^{(G)})$$

$$DE/best/1: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{best}^{(G)} + F\cdot(\mathbf{x}_{r_1}^{(G)} - \mathbf{x}_{r_2}^{(G)})$$

$$DE/current\ to\ best/1: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_i^{(G)} + F\cdot(\mathbf{x}_{best}^{(G)} - \mathbf{x}_i^{(G)}) + F\cdot(\mathbf{x}_{r_1}^{(G)} - \mathbf{x}_{r_2}^{(G)})$$

$$DE/best/2: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{best}^{(G)} + F\cdot(\mathbf{x}_{r_1}^{(G)} - \mathbf{x}_{r_2}^{(G)}) + F\cdot(\mathbf{x}_{r_3}^{(G)} - \mathbf{x}_{r_4}^{(G)})$$

$$DE/rand/2: \quad \mathbf{v}_i^{(G)} = \mathbf{x}_{r_1}^{(G)} + F\cdot(\mathbf{x}_{r_2}^{(G)} - \mathbf{x}_{r_3}^{(G)}) + F\cdot(\mathbf{x}_{r_4}^{(G)} - \mathbf{x}_{r_5}^{(G)})$$

3.3 Crossover /Recombination

After mutation, a 'binary' crossover operation forms the trial vector $u_i^{(G)}$ according to the target vector $x_i^{(G)}$ and its corresponding mutant vector $v_i^{(G)}$

$$u_{i,j}^{(G)} = \begin{cases} v_{i,j}^{(G)} & \text{if } rand(0,1) \le CR \text{ or } j = j_{rand}, \\ x_{i,j}^{(G)} & \text{otherwise} \end{cases} \qquad (7)$$

$$i = 1,2,\ldots,NP \text{ and } j = 1,2,\ldots,D.$$

3.4 Selection

The DE algorithm uses a greedy selection. The selection operator selects between the target and corresponding trial vectors. A member of the next generation becomes the fittest vector, i.e. vector with the better fitness value. The selection rule of minimization problems is:

$$\mathbf{x}_i^{(G+1)} = \begin{cases} \mathbf{u}_i^{(G)} & \text{if } f(\mathbf{u}_i^{(G)}) \le f(\mathbf{x}_i^{(G)}), \\ \mathbf{x}_i^{(G)} & \text{otherwise.} \end{cases} \qquad (8)$$

4 An Alternative Differential Evolution Algorithm:

An Alternative Differential Evolution algorithm is given. The main changes are:

1- The scale factor F will be consider as a dynamic factor between a lower bound chosen as 0.3 and upper bound 0.7 and satisfying the following equation

F (scale factor) =(A(1)+A(3)-A(2))*0.1/(A(3)+(A(2)-A(1)))

Where A=random permutation (Population Size).

2- The crossover rate will be taken as fixed value =0.3

4.1 Experimental Results

The Parameters that used in an alternative differential evolution algorithm: The initial population was generated uniformly at random in the range, as specified in Table (1)

***TABLE 1:** Benchmark functions*

	Test Function	Dimension	Range	Minimum Value
f1	Sphere	30	(-100,100)	0
f2	Rosenbrock	30	(-30,30)	0
f3	SCHWEFEL	30	(-500,500)	-12569.5
f4	Rastrigin	30	(-5.12,5.12)	0
f5	ACKLEY	30	(-32,32)	0
f6	GRIEWANK	30	(-600,600)	0
f7	Goldstein & Price Function (gold)	2	(-2.2)	3
f8	Beale	2	(-4.5,4.5)	0
f9	Booth	2	(-10,10)	0
f10	Easom	2	(-100,100)	-1
f11	Hartmann	3	(1,3)	-0.30048
f12	Hump	2	(-5,5)	0
f13	Sum Squares	N	(-10,10)	0
f14	levy	N	(-10,10)	0
f15	Powell	N	(-4,5)	0

Throughout this paper there are three comparisons:

1. The first Comparison between Alternative DE and Self-Adaptive DE the results for the benchmark problems f1–f15 are shown in Table (2), we have used F =0.5,CR=0.9and for the (original) DE algorithm. Our decision for using those values is based on proposed values from literature R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," (1997) Both the Alternative differential evolution and DE standard are tested with 30 independent runs for each test function. The population size is set to 50. The implementation is done using Matlab version R2016a.

***TABLE 2:** The experimental Results for the proposed algorithm and the standard differential evolution algorithm on15 benchmarks problems*

	Standard D.E	Alternative differential evolution	Minimum Value
f(1)	1.19E-22	1.23E-34	0
f(2)	0	0	0
f(3)	-6181.6892	-11731.5208	-12569.5
f(4)	0	0	0
f(5)	8.8818e-16	8.8818e-16	0
f(6)	0	0	0
f(7)	3	3	3
f(8)	0	0	0
f(9)	0	0	0
f(10)	-1	-1	-1
f(11)	-0.30048	-0.30048	-0.30048
f(12)	4.65E-08	4.65E-08	0
f(13)	-7.54E+03	6.80E-27	0
f(14)	3.70E-21	2.07E-20	0
f(15)	2.33E-13	0.0050262	0

2-The second Comparison between Alternative DE and Self-Adaptive DE the results for the benchmark problems f1–f7 are shown in Table (3); both the Alternative differential evolution and self-adaptive DE are tested with 50 independent runs for each test function. The population size is set to 50. The implementation is done using Matlab version R2016a.

TABLE 3: *THE EXPEREMINTAL RESULTS FOR PROPOSED ALGORITHM AND SELF-ADAPTIVE DE (JDE) ON SEVEN BENCHMARKS PROBLEMS*

	Self-AdaptiveDE	**Alternative differential evolution**	**Minimum Value**
f(1)	1.1 E-28	1.8925e-58	0
f(2)	1.0E-23	0	0
f(3)	3.1E--14	-7304.8153	-12569.5
f(4)	0	0	0
f(5)	0	0.003127	0
f(6)	0	0	0
f(7)	3.15 E-3	3	3

3-The Third Comparison between Alternative DE and four algorithms: Static Heterogeneous DE (sHDE), Dynamic Heterogeneous DE (dHDE), DE with neighborhood search(NSDE), DE using neighborhood-based mutation (DEGL), the results for the benchmark problems f(2), f(5), f(6) are shown in Table (4)

TABLE 4: *THE EXPERIMENTAL RESULTS FOR THE PROPOSED ALGORITHM AND EIGHT OTHER DIFFERENTIAL EVOLUTION ALGORITHMS ON THREE BENCHMARKS PROBLEMS*

	f(2)	f(5)	f(6)
sHDE	1.11E − 29	4.14E − 15	0.00E + 00
dHDE	1.73E − 29	4.14E − 15	0.00E + 00
NSDE	2.65E − 25	5.97E − 10	7.93E − 26
DEGL	6.89E − 25	5.98E − 23	2.99E − 36
ADE	0	8.88E-16	0

4 CONCLUSION

[1] An alternative differential evolution algorithm is developed .the proposed algorithm shows the speed of convergences of the proposed algorithm is better than that of the standard evolution algorithm.
[2] Compared with other algorithms, the proposed 1 arrives as the exact solution in some cases and better results and other cases will compared with 10 well known differential evolution algorithms.
[3] Additional work will be carried out to improve the proposed algorithm.

REFERENCES

[1] A. Cichoń, E. Szlachcic and I. F. Kotowski, Differential evolution for multi-objective optimization with self-adaptation, Proceedings of the 14th international conference on Intelligent engineering systems, (2010).

[2] A. Musrrat, P. Millie, A. Ajith, Improved differential evolution algorithm with decentralization of population, International Journal of Bio-Inspired Computation, November (2011).

[3] A. Slowik, and M. Bialko, Training of Artificial Neural Networks using Differential Evolution Algorithm, Poland, May (2008).

[4] E.K. BURKE and G.KENDALL, Search Methodologies, Springer (2005).

[5] E. K. da Silva, H. J. C. Barbosa and A. C. C. Lemonge, An Adaptive Constraint Handling Technique for Differential Evolution in Engineering Optimization, International Conference on Engineering Optimization, June (2008)

[6] J. Lampinen Multi-Constrained Nonlinear Optimization by the differential evolution Algorithm, Finland, (2001).

[7] J. Teo and M.Y.Hamid A Parameterless Differential Evolution Optimizer, Proceedings of the 5th WSEAS/IASME Int. Conf. on Systems Theory and Scientific Computation, Malta, September, (2005).

[8] J. TVRD´ IK, Differential Evolution with competitive setting of control parameters, January (2007).

[9] R. Sarker, M. Mohammadian and X. Yao, Evolutionary Optimization, Kluwer Academic Publeshers (2003)

[10] R. STORN, K. PRICE Differential Evolution – A simple and Efficient Heuristic for global optimization over continues spaces, (1997).

[11] S. Rahnamayan, H. R. Tizhoosh and S. Magdy Opposition-based differential evolution for optimization of noisy problems, July (2006).

[12] S. Sumathi, T. Hamsapriya and P. Surekha, Evolutionary Intelligence, Springer (2008).

[13] T. Weise, Global Optimization Algorithms Theory and Application, July (2007).

[14] T. W. Liao, Two Hybrid Differential Evolution Algorithms for Engineering Design optimization, (2010).

[15] U.K. Chakraborty, Advanced in differential evolution, Springer (2008.)

[16] V. Feoktistov, Differential Evolution in search of solutions. Springer, New York, (2006).

[17] Z. Yang, K. Tang and X. Yao, Differential Evolution for High-Dimensional Function Optimization (2007).