PRIMARY RESEARCH

Two enhanced differential evaluation algorithms for expensive optimization

Muneer Maaroof Hasan [1], Oğuz Altun [2, *]

[1, 2] Computer Engineering Department, Yildiz Technical University, Istanbul, Turkey

Index Terms
Deferential Evaluation
Expensive Optimization
Single Objective Optimization

Received: 22 April 2016
Accepted: 12 May 2016
Published: 24 June 2016

Abstract—In this paper, the meta-heuristic algorithm which named Differential Evaluation (DE) has been improved. The improving made to increase the exploration rate and decrease the run time. Since DE needs too long time, when we implement it to solve computational expensive problems, we developed two different versions of DE named by Enhanced1 Differential Evaluation (E1DE) and Enhanced2 Differential Evaluation (E2DE). E1DE and E2DE were introduced to solve Computationally Expensive Optimization (CEO). Problems discussed and tested using all 15 test functions of the Special Session & Competition on Real-Parameter Single Objective Optimization (Expensive Case) at Congress on Evolutionary Computation 2015 (CEC-2015). The results show that the work significantly improved the basic DE in time by 54% and in results by 86%.

© 2016 The Author(s). Published by TAF Publishing.

I. INTRODUCTION

Optimization problem sometimes requires computationally expensive simulations for calculating its candidate solutions [1]. For such problems, the algorithm has a far fewer function evaluation budget. We develop two improvements of DE that makes it better for these kinds of expensive optimization problems. The first improvement is simply by increasing the number of mutation operation, which creates a tweaked individual, to two operations with different individuals so there would be two tweaked individuals for every selected candidate out of the population. In this case, we increase the exploration rate by finding the best individuals to create the tweaked one so surely it will decrease the time for convergence. While the second improvement is on the way that the DE selects the individuals for the mutation operation, we made the selection like a wheel, to create tweaked individual as result all the individuals in the population will participate in creating the new generation.

Many functions have been used to evaluate the candidates of a problem with insufficient results, especially when the candidates are a binary vector, so there are many functions have been invented to solve that kind of problems. The CEC 2015 benchmark is one of those solutions [1]. We tested our algorithms on the all the 15 test functions of the mature test bed Congress on Evolutionary Computation 2015.

The performance of DE is controlled by two parameters, mutation factor (or called amplification weight factor) and crossover probability (or called Crossover rate) [8]. Moreover, when DE explores a region of space, it would require using parameters values that different from another region of space for high efficiency of search [9]. There are manyresearches about how to set the

*Corresponding author: Oğuz Altun
E-mail: oguz211@gmail.com

© The Author(s). Published by TAF Publishing. This is an Open Access article distributed under a Creative Commons Attribution-NonCommercialNoDerivatives 4.0 International License

parameters for DE but till now there is no distinct relation between search space and settings of DE's parameters [10]

At some points, the DE stops generating, or we should say stop tweaking its population's individuals, and this situation called stagnation [11]. These problems have been studied in self-adaptive region, solved by letting the parameters automatically change its value to get out of the local optimum solution [12]. Our work is one of these kind of solutions, since it can escape from the local optimum by selecting the individuals to generate different offspring, and the results showed that there are good results with our work.

II. DIFFERENTIAL EVALUATION

A. *Basic Differential Evaluation*

Differential Evolution (DE) is a meta-heuristic algorithm for continuous functions [2]. DE was introduced in 1995 by [3]. It was used by many researchers to solve different problems [2]. The generation (the population) updates its individuals in every step, with a tweaked individual that has higher or lower value of the fitness function and that depends on the problem that we deal with. If we were trying to minimize then we will select the individual which has the lowest fitness value. While if we were trying to maximize then we will select the individual which has the highest fitness value [8]. In each iteration of the DE algorithm, the tweaked individual will be generated by using two operations. The first one is Mutation and the second one is Crossover. DE compares the tweaked individual with the current original individual according to the fitness function value and then replace it with the original one (its parent), if it was better, or throw it away if it was worst.

In Mutation, the algorithm selects three vectors out of the population randomly then use one of three schemes defined in DE, DE/rand/bin is one of the popular schemes, to produce a new sample vector (the tweaked vector) [7]. While the Crossover takes that new sample from the Mutation as an input and uses some tweaking on its values [7].

The Mutation operation controlled by an amplification weight factor (a vector lies between 1 and 0). The formula (1) shows the DE/rand/bin Mutation schemes.

$$X_{i,G} = V_{r1,G} + F\,(V_{r2,G} - V_{r3,G}) \qquad (1)$$

Where X is the tweaked individual after the mutation operation, $V_{\mathbf{r1,G}}$, $V_{\mathbf{r2,G}}$ and $V_{\mathbf{r3,G}}$ are vectorschosen randomly out of the population and F is an amplification weight factor.While the Crossover operation controlled by a parameter called Crossover rate (**Cr**), the Crossover operation is simply a probability kind of operations as we see in formula (2).

$$T(i,i,G) = \begin{cases} X(j,i,G) & \text{if randi } [1,0] \le Cr \\ V(j,i,G) & \text{otherwise} \end{cases} \qquad (2)$$

Where T is the tweaked individual after the crossover operation, X is the output of the mutation operation and V is the original vector (out of the population) that we trying to generate child by implementing the mutation and the crossover on it. And where rand [1,0] is an inbuilt function in MATLAB that generates random numbers between 0 and 1. Figure 1 shows briefly the main steps of the basic DE.

B. *Enhanced 1 Differential Evaluation*

Since the powerful of DE is related to its way to mutate the individuals of its population, the mutation process is the core of DE. The enhancing in E1DE and in E2DE has been made on the mutation process to do a fast convergence as result increasing the exploration. In E1DE the mutation process in each step will take place twice, first one will be by selecting the three individuals randomly and the second one will be by selecting the inverseof the three individuals in first mutation operation and that will produce two different tweaked individuals. After the two tweaked individuals passed throw the crossover operation the E1DE will select the best one and compare it with the current original individual. As usual, it will replace the original one if it was best and throw it away if it was worse.

The first mutation process will follow the original schemes, for example, DE/rand/bin, but the second one will depend on the first one to select the individuals out of the population so the order of these two mutation operations should be consecutive. Figure 2 shows the E1DE and how the two mutation operations take place in the main procedure.

C. *Enhanced 2 Differential Evaluation*

In standard DE, the selection of the three individuals for mutation is randomly and in this way, there is a probability that a good individual will never be chosen to create the tweaked individuals, as we name it, so the

ISSN: 2414-3103
DOI: 10.20474/japs-2.2.4

TABLE 1
THE RESULT OF IMPLEMENTING 10D PROBLEM TO TEST THE THREE ALGORITHMS

function	Algorithm name	Mean	Standard deviation	Median	Time in Sec
f1	Standard DE	22446772180	31021496.15	22431852764	**127.5531295**
	Enhanced1 DE	22450708007	72847546.38	22431852764	188.0954066
	Enhanced2 DE	**22438588158**	20423329.53	22431852764	127.6096615
f2	Standard DE	298263865.1	1668699.732	297555022	**127.7088035**
	Enhanced1 DE	**297909103.6**	1579180.99	297555022	187.704087
	Enhanced2 DE	298116632.4	2164652.333	297555022	128.2111776
f3	Standard DE	313.6570523	0.127102445	313.6657794	**170.5774288**
	Enhanced1 DE	313.706152	0.116351623	313.6703955	291.0310504
	Enhanced2 DE	**313.6314059**	0.17505751	313.6105995	185.0814753
f4	Standard DE	**2731.109382**	37.67265812	2718.720383	140.9805855
	Enhanced1 DE	2733.581094	39.13305349	2718.720381	204.895086
	Enhanced2 DE	2739.712717	73.84419002	2718.720381	**132.2451977**
f5	Standard DE	500.1107523	0.094519534	500.0985657	**159.5496698**
	Enhanced1 DE	500.1263294	0.105692179	500.1034332	251.1457535
	Enhanced2 DE	**500.091855**	0.074999911	500.0828008	167.9505809
f6	Standard DE	604.9009223	0.007909668	604.8987479	137.349538
	Enhanced1 DE	604.89931	0.001188355	604.8987488	191.8612599
	Enhanced2 DE	**604.8996766**	0.002265901	604.8987506	**127.7683108**
f7	Standard DE	758.1597691	0.139998711	758.1140656	128.274604
	Enhanced1 DE	758.129514	0.035195214	758.1140656	188.8563623
	Enhanced2 DE	**758.122248**	0.023770007	758.1140656	**127.9998958**
f8	Standard DE	**204003.091**	582.7046966	203709.0093	**128.7405082**
	Enhanced1 DE	204031.4731	648.370023	203709.0093	188.70292
	Enhanced2 DE	204866.9768	3188.053791	203709.0133	129.1010406
f9	Standard DE	903.8772	0.041115957	903.8861879	130.3811113
	Enhanced1 DE	903.9879458	0.082415715	903.9547187	194.8995639
	Enhanced2 DE	**903.8769023**	0.031546024	903.8693103	**130.1528177**
f10	Standard DE	2031397153	3115975.309	2029671379	133.4868069
	Enhanced1 DE	2031770295	3875108.679	2029670198	196.6555167
	Enhanced2 DE	**2030603070**	2618365.605	2029670200	**133.3574468**
f11	Standard DE	1477.376043	2.096880487	1476.173229	**140.872317**
	Enhanced1 DE	**1476.597528**	1.127580081	1476.189151	216.0664078
	Enhanced2 DE	1476.992297	1.443781794	1476.189251	140.9173593
f12	Standard DE	28669.5882	227.4029358	28554.60717	136.398957
	Enhanced1 DE	**28557.12782**	9.25141382	28554.60712	202.7311281
	Enhanced2 DE	28687.10909	305.0033901	28554.60712	**136.1693277**
f13	Standard DE	3711.482007	0.374756408	3711.318756	138.6885592
	Enhanced1 DE	3712.364929	5.562198011	3711.318756	207.8094264
	Enhanced2 DE	**3711.3774**	0.17941681	3711.318756	**138.5885526**
f14	Standard DE	1651.340525	1.107264534	1650.738022	**137.9278771**
	Enhanced1 DE	**1651.288388**	0.693239759	1650.737994	206.5472122
	Enhanced2 DE	1651.380138	1.075252271	1650.738897	138.0381629
f15	Standard DE	2155.539978	3.086328105	2154.928171	**183.7776149**
	Enhanced1 DE	2154.870334	2.914348869	2153.25722	301.8572172
	Enhanced2 DE	**2154.338523**	2.459172143	2152.873318	184.0005855

```
Differential Evolution Algorithm
• Generate the initial population (with random values in boundaries)
• For each generation
    o For each individual in the population
        ▪ Apply the mutation scheme to select three individuals of the population
        ▪ Apply the crossover operation on the output of the mutation process
        ▪ Replace the current individual with the child, if the child is better
    o Update the population with the accepted individuals
• Return the best individual from the population at hand.
```

Fig. 1. Pseudo-code of standard DE

```
E1DE algorithm
• Generate the initial population (with random values between any boundaries)
• For each generation
    o For each individual in the population>
        ▪ Apply the first mutation scheme to select three individuals to create a tweaked individual
        ▪ Apply the second mutation, based on the first mutation by selecting the three individuals, to
          generate another tweaked individual
        ▪ Apply the crossover operation on both first and second output of the mutation process
        ▪ Replace the current individual with the best child out of the two children, if the child is better
    o Update the population with the accepted individuals
• Return the best individual in the last generation
```

Fig. 2. Pseudo-code of the E1DE algorithm

```
E2DE algorithm
• Generate the initial population (with random values between any boundaries)
• For each generation
    o For each individual in the population>
        ▪ Set a counter (X) to 1,  X=1
        ▪ POP(X),POP(X+1) and POP(X+2)  are the three individuals of the population for the mutation
          operation consecutively
        ▪ If X >= POP. Size then X=1
        ▪ Apply the crossover operation on the output of the mutation process
        ▪ Replace the current individual with the child, if the child is better
    o Update the population with the accepted individuals
• Return the best individual in the last population
```

Fig. 3. Pseudo-code of the E2DE algorithm

change has been done on selecting the three individuals for mutation. Simply the way that E2DE selects the three individuals will be like a wheel. It will select the first three individuals for the first iteration and then select the second three individuals for the second iteration….till it reaches the last three of the population then it starts from the beginning and figure 3 shows how this operation has been done by using a counter (variable X). In this way, the good individual should be selected one time at least and that is how the time got decreased because the optimal solution will be around the best value we have got.

The Mutation operation is still controlled by an amplification weight factor (a vector lies between 1 and 0). The formula (4) shows the new Mutation schemes.

$$\begin{cases} X(i,G) = V(i,G) + F * \left(V\big((i+1),G\big) - V\big((i+2),G\big)\right) \\ X\big((i+1),G\big) = V\big((i+3),G\big) + F * \left(V\big((i+4),G\big) - V\big((i+5),G\big)\right) \\ X(n,G) = V\big((n-3),G\big) + F * \left(V\big((n-2),G\big) - V\big((n),G\big)\right) \end{cases}$$

Where X is the tweaked individual after the mutation operation, Vis a vector chosen randomly out of the population, F is an amplification weight factor and n is the number of individual in our population.

III. BENCHMARK AND EXPERIMENTAL SETTINGS

A. *Algorithms Parameters*

DE, E1DE and E2DE is tested by using all fifteen functions of the Congress on Evolutionary Computation 2015 with 10D problems, 50 individuals in the population, crossover rate (Cr) fixed to 0.7, Function Evaluations (FES) are fixed to 10000 and independent runs are fixed to 30. The random numbers were generated by using the inbuilt unifrnd(Min,Max,Size) function in R2015a MATLAB.

- PC Configurations

Operation system: windows 10 Home
Processor (CPU): Intel® Pentium® CPU B940 @ 2.00GHZ (2 CPUs)
Memory: 4000MB RAM

IV. RESULTS AND DISCUSSION

The performance of the two enhanced DE version has been tested and compared with original DE on fifteen computationally expensive single objective functions which are presented in the Congress on Evolutionary Computation 2015. Two functions of them are unimodal, seven of them are simple multimodal, three of them are hybrid and the rest three of them are composition. We tested our work by setting the benchmark parameters to ten dimensions only (10D) according to the limited space on memory that we have and with thousand function evaluation times. The results compare the best, standard deviation, median, and the time of the three algorithms as shown in Table (1).

In the table (1), since the problems are minimization problems, the algorithm with the lowest mean value is better than others. Non algorithm seems to be the best one all the time but here is the fact. When we care about the time more than the result then we can select the enhanced2 differential evaluation algorithm so far. But when we care about the result more than the time then we can select the enhanced1 differential evaluation algorithm.

In Unimodal functions (f1 and f2), the time was on the standard's side, while the mean value was separated between E1DE and D1DE. But for Unimodal functions (f3, f4, f5, f6, f7, f8 and f9), mean values were all for E2DE and the time was separated between DE and E2DE, that makes E2DE more useful since it has better values and time with many functions. While with Hybrid functions (f10, f11 and f12), the DE has never shown neither in best value of mean nor in time, it looks that E1DEhas the preference with mean value and E2DE in time. Finally, for Composition functions (f13, f14 and f15), the time and mean values are totally distributedbetween the three functions but f13 has ahigh priority since it is best in time and in mean value. Overall, E2DE seems to be the best one of the three functions, it has thebest mean and less computation time most the time.

V. CONCLUSIONS AND FUTURE WORKS

DE is a strong and simple meta-heuristic algorithm for continuous numeric search domain and it can reach the optimal solution very fast and faster than the other algorithm as we have seen when we tried to implement other algorithms. DE can reach the optimal solution at the fifty iteration, or maybe less than fifty, and that because the way that DE searches and mutates the candidates (individuals), so we will not need to iterate the DE more than hundred times when we want to implement it on any problem, like feature selection for example, and we can change its parameters to make it work as we want.

For future works, the initialization of the DE population could be changed so it will not be random and as result it will reach the optimal solution in less time or we can do a hybrid DE by using any stochastic algorithms like Hill climbing or Tabu search algorithms to find the optimal vectors of any domain then use that vectors as a population initialization for DE.

E1DE and E2DE, as we have seen in result section, are good improved Differential Evaluation versions. One of them, E2DE, efficiently reached the optimal solution with taking short evaluation time. That makes E1DE a great improvement on the basic DE. However, E1DE reached the optimal solution when both DE and E2DE were not able to.

REFERENCES

[1] Q. Chen, B. Liu, Q. Zhang, J. J. Liang, P. N. Suganthan and B. Y. Qu. "*Problem definitions and evaluation criteria for CEC 2015 special session on bound constrained single-objective computationally expensive numerical*

optimization", Computational Intelligence Laboratory., Zhengzhou, CN, Tech. Rep. 201212 2014

[2] S. Luke, "*Essentials of Metaheuristics*. Virginia, VA: Lulu Com, 2013.

[3] R. Storn and K. Price, "Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, 341-359, 1997. **DOI:** 10.1023/A:1008202821328

[4] P. T. Bharathi and P. Subashini, "Optimal feature subset selection using differential evolution with sequential extreme learning machine for river ice images," in *TENCON 2015-2015 IEEE Region 10 Conference*, 2015, pp. 1-6.

[5] B. Xue, W. Fu and M. Zhang, "Differential evolution (de) for multi-objective feature selection in classification," in *Proceedings of the Companion Publication of the Annual Conference on Genetic and Evolutionary Computation ACM*, 2014, pp. 83-84. **DOI:** 10.1145/2598394.2598493

[6] S. Gadat and L. Younes, "A stochastic algorithm for feature selection in pattern recognition," *Journal of Machine Learning Research*, vol. 8, no. 3, pp. 509-547, 2007.

[7] A. S. Poonia, T. K. Sharma, S. Sharma and J. Rajpurohit, "Aesthetic differential evolution algorithm for solving computationally expensive optimization problems," in *Advances in Nature and Biologically Inspired Computing.* Berlin. Germany, Springer International Publishing, 2016, pp. 87-96.

[8] R. Gämperle, S. D. Müller and P. Koumoutsakos, "A parameter study for differential evolution," in *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, Berlin. Germany, Springer International Publishing, 2002, 293-298.

[9] J. Ronkkonen, S. Kukkonen and K. V. Price, "Real-parameter optimization with differential evolution," in *Proceding IEEE CEC*, 2005, pp. 506-513.

[10] E. Mezura-Montes, J. Velázquez-Reyes and C. A. Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 8th Annual conference on Genetic and Evolutionary Computation,* ACM, 2006, pp. 485-492. **DOI:** 10.1145/1143997.1144086

[11] S. M. Guo, C. C. Yang, P. H. Hsu and J. S. H. Tsai, "Improving differential evolution with a successful-parent-selecting framework," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 717-730, 2015. **DOI:** 10.1109/TEVC.2014.2375933

[12] Y. Lou, J. Li and G. Li, "A differential evolution algorithm based on individual-sorting and individual-sampling strategies," *Journal of Computational Information Systems*, vol. 8, no. 2, pp. 717-725, 2012.

— This article does not have any appendix. —