2009 International Conference on Machine Learning and Computing
IPCSIT vol.3 (2011) © (2011) IACSIT Press, Singapore

Improving the Performance of Differential Evolution Algorithm with Modified Mutation Factor

Ching-Wei Chien [1], Zhan-Rong Hsu [1], Wei-Ping Lee [1 +]

[1] Chung Yuan Christian University, Information Management Department, Chun li, Taiwan, 32023.

Abstract. Differential Evolution (DE) algorithm [1, 2], first published in 1995, has proven to be a powerful tool for complicated optimization problems. Its outstanding performance and accuracy makes it applicable to different research fields. However, since differential evolution belongs to the Evolutionary Algorithm, problems like dropping in local optimum may also occur in this algorithm. In order to improve the performance of differential evolution, we propose a novel algorithm, which will generate a dynamical function for changing the differential evolution parameter "mutation factor" replace traditional differential evolution algorithm use constant mutation factor. The present study is to increase the performance of this novel algorithm and to avoid dropping into local optimum.

Keywords: Differential Evolution(DE), Evolutionary Algorithm(EA), Mutation Factor, Modified Mutation Factor Differential Evolution(MFDE).

1. Introduction

Evolution Computation (EC) is one of the powerful tools used to solve complex problems (Scheduling, routine planning, data mining, resource allocation, and artificial intelligence), which cannot be solved with traditional mathematical methods. Differential algorithm, one of the evolution computations, possesses the same drawback in dealing high-dimension complex function cannot break away the local optimum to result in convergence.(Repetition) This study attempts to improve the “mutation factor” parameter of the set with a view to increase solution accuracy and to enhance the ability to escape from the regional optima.

2. Literature Review

2.1. Differential evolution (DE)

Differential evolution, one of the evolutionary computations, was first published in 1995 by Storn and Price. The Differential evolution consists of the simulation annealing mechanism, but the authors in 1996 published a modified version using floating-point computing to replace bit computing, and removed the simulation annealing mechanism to turn into familiar differential evolution.

Differential evolution is an “uncomplicated program”, “easy to implement”, “simple in structure” and has “fewer parameters to setup” [3].

The main concept of differential evolution is the added difference between the individual. In real world, the difference between personal is the competition advantage, rely on continuous learning and imitate another’s competition advantage to promote the civilization keep evolution.

During a generation, differential evolution selects the solution vector and acquires their difference and multiplies a mutation factor to obtain a new vector called “donor vector”. Furthermore it adds the donor vector to another solution vector to become trial vector.

[+] Corresponding author. Tel.: +886956881654 ; fax: +88632655499.
E-mail address:robler.jane@gmail.com.

This order can help distributing the solution vector to avoid excessive convergence to local optima

The procedure of Differential evolution as shown in Fig 1, and parameter will introduces in table 1.

Table 1
Parameter of Differential Evolution

Population Numbers(*NP*)	The size of population decides the total number of the solution vectors in the same iteration. Large population required longer computing time, but too small population in a search of a large solution.
Mutation Factor(*F*)	This factor decides how many perturbation ratios the solution can acquire. If the value is greater, the magnitude of jump will be increase. In other words, population can break away the regional optimum effectively. Small mutation factor can converge rapidly, but has a high probability of dropping in a regional optimum.
Crossover rate(*CR*)	Crossover rate decides the swap probability between trial vector and target vector.

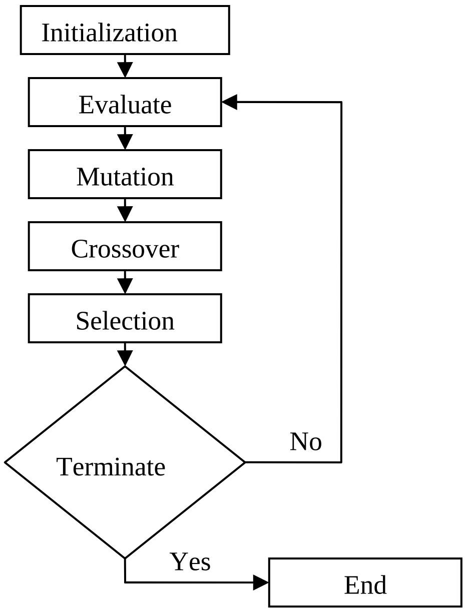


Fig. 1 The procedure of Differential evolution

Initialization:

The main assignment is to decide the above parameter, and to ensure the initial value of solution vectors control in restriction scope to make sure solution's rationality.

Mutation:

Selected several solution vector randomly, and acquire the difference between the vectors to multiply mutation faction furthermore added on target vector to assist target vector mutate. There are two traditional common mutation types [4]:

DE/rand： $v_{i,g+1} = x_{ri,g} + F(x_{r2,g} - x_{r3,g})$

DE/best： $v_{i,g+1} = x_{best} + F(x_{r2,g} - x_{r3,g})$

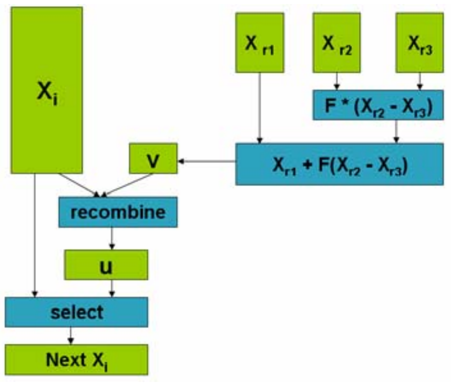


Fig. 2 The diagram of differential evolution

Crossover:

Upon approaching the mutation operation, the donor vector will swap the information with the target vector (x_i) randomly. After crossover, a trial vector u will be produced. The following formulation will be used to decide whether the component *i* is composed from target vector x_i or donor vector v_i in the generation *j*.

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & \text{if } rand \le CR \\ x_{j,i,G} & \text{if } rand > CR \end{cases}$$

Here, *rand* is a random number that obey uniform distribution between 0 and 1. *CR* records the crossover rate.

Selection:

After the mutation and crossover operations, the trial vector and target vector will approach to fitness functions to determine the one to be reserved for the next generation.

2.2. Relative research of mutation factor

Differential evolution is a significantly faster optimization algorithm, but it may probably result in dropping in regional optimum like else evolutionary computation such as Particle Swarm Optimization (PSO) and Genetic Algorithm. Mutation factor (F) is a decisive factor to decide the ability of exploration (Global search) and exploitation (local search) [5]. Early studies main though experience or trial and error to find a suitable value. In earliest version, these values are (0, 2] and [1, 2]. But the original authors found that the value of F in the range of 0.4 to 1 was an interval that could be considered, and recommend 0.5 as a good initial value [6]. Scholars have suggested that the scope of F value should be set to [0.4, 0.8], and it is still suitable for use with some functions of the settings, rather than the common optimal value [7].

Kaelo and Ali propose a random F value between [-1,-0.4] and [0.4, 1] [8]. Some scholars attempt to dynamically compute F value in every generation.

This study proposes a new function to produce a dynamical F value providing appropriate consideration to exploration and exploitation. In most cases, this novel method can provide better solution.

2.3. Modified mutation Factor Differential Evolution

In this section, we propose a modified mutation factor differential evolution (MFDE) to improve traditional DE cannot to give consideration to exploration and exploitation.

The following is procedure of DE shown in Table 2:

Table 2
Procedure of Differential Evolution

Step 1：	Initialization. Include the total number of solution vector; Maximum of generation; Mutation Factor; Crossover rate; define the solution space between the lower bound and upper bound. Initialize of each variable in its own range.
Step 2：	Generated initial solution vector randomly.
Step 3：	Evaluate the fitness value of the solution vector. If the fitness value doesn’t improve, import deceleration factor into mutation. On the other hand, import acceleration factor upon improving the fitness value.
Step 4：	Mutation. Select several variable vectors and acquire their difference and multiply the *F* value from the *F* value function. Produce donor vector at the end.
Step 5：	Recombination. Swap donor vector and target vector by *CR* to produce the trial vector.
Step 6：	Selection. Compare target vector and trial vector to determine the one can be reserved.
Step 7：	If the termination conditions are unsatisfactory, go back to Step 2, or output the optimal solution if the termination conditions are

This study uses the combination of linear decrease and random mutation factor. Each generation will produce a new mutation factor value by following function:

$$F = s * \sqrt{r(0,1)^2 * d} - b$$

Each parameter will descript below:

Linear decreasing factor(d)：	During the generation, the value of “Linear decrease factor” will gradually decrease from 1.2 to 0.2, but not 0. In order to avoid *F* becoming 0, in an early stage, linear decreasing function allows the algorithm to have more variation capacity to explore all the search spaces, and increases the ability for exploration. In a later period will decreasing the variation capacity to focus on local search for regional development.
Random variable (r)：	When the mutation factor only employs the linear decreasing factor, then it will cause the algorithm to converge prematurely to make the performance lower in complex multi-modality function. Therefore combining the random variable can let the *F* value fluctuate with probability, giving a certain degree of probability to the breakaway region optimum. Moreover it also increases the regional development capacity. In this study, $r(x, y)$ represents a random variable that obey uniform distribution between x and y.
Acceleration factor (s)：	To provide consideration to balance global exploration and local exploitation. During the generation, the fitness value has relative improvement, reduce *F* value to let algorithm acquire more exploitation capacity increasing the speed of convergence.
Deceleration factor(b)：	In contrast with the acceleration factor, during the generation, the algorithm still stagnate and the fitness value cannot be improved further, giving larger *F* value to gain in global exploration capacity prevent dropping in region optimum.

3. Research Methods and Experiment Design

3.1. Experimental Design

This study proposes the MFDE algorithm that is constructed on traditional DE/rand, utilizing dynamic generating function to improve the original structure. Basically, this preserves most characteristic of DE/rand.

3.2. Parameter setup

In this study, unless a particular parameter setup is specified, parameter setup that shown in table 3 and 4 will be accepted.

In our algorithm MFDE, the F value will generate by dynamic generate function. DE/rand and DE/best accept constant 0.5 to be the mutation factor value.

The parameter setting of DE and MFDE has shown in table 3 and 4:

Table 3
Parameter Setup of DE

Mutation Factor(*F*)	0.5
Crossover rate (*CR*)	0.9
Upper bound	100
Low bound	-100
Number of vector(*NP*)	50
Dimension	30/60
Maximum of generation	1000/2000

Table 4
Parameter Setup of MFDE

Linear decreasing factor (d)	1~0
Random variable(r)	rand(0,1)
Acceleration factor (s)	1.5
Deceleration factor (b)	0.2

3.3. Benchmark Function

This study utilize four popular benchmark functions for evaluate the modified algorithm performance. All of the benchmark will be examine repeat 1000 times and evaluate the average, moreover compare with DE/rand and DE/best for examination MFDE's effectiveness.

Function characteristics displayed as Table 5.

All the benchmark theoretical minimum value is 0. Functions and its' category shown as table 5:

TABLE 5
Benchmark Function

No.	Name	Category	Function
f1	Sphere	Single modality	$f(x)=\sum (x_i)^2$
f2	Rosenbrock	Single modality	$f(x)=\sum [100*(x_i^2-x_{i+1})^2+(1-x_i)^2]$
f3	Griewank	Multi modality	$f(x)=\sum \frac{x_i^2}{4000}-\Pi\cos(\frac{x_i}{\sqrt{i}})+1$
f4	Rastrigrin	Multi- modality	$f(x)=\sum (x_i^2-10*\cos(2\pi x_i)+10)$

4. Experimental Results

This section will compare the performance of MFDE, DE/rand, and DE/best. 30-dimension executes 1000 generations and 60-dimension executes 2000 generations .All experimental result compared in Table 6 to 10.

After utilize the MF mechanism in traditional DE/rand Algorithm, the effectiveness of solving the function Sphere has significantly improvement. DE/best show the high performance in single modality function. Fig. 3 and 4 represents the convergence progress of function Sphere.

Table 6
Experimental result of Sphere

Function (f1)	Dim.	Avg.	Best
DE/rand	30	3.6450e-10	1.3824e-10
	60	1.0413e-11	6.7675e-12
DE/best	30	3.0378e-16	9.7063e-17
	60	1.7857e-19	5.4536e-20
MFDE	30	3.1387e-12	8.6077e-13
	60	6.4354e-16	4.4541e-17

Table 7
Experimental result of Rosenbrock

Function (f1)	Dim.	Avg.	Best
DE/rand	30	1.1963e+02	3.9919e+01
	60	1.2521e+02	6.9805e+01
DE/best	30	7.1773e+01	6.4050e+00
	60	1.3764e+02	2.6281e+01
MFDE	30	1.1869e+02	3.3556e+01
	60	1.2734e+02	5.2708e+01

Table 8
Experimental result of Griewank

Function (f1)	Dim.	Avg.	Best
DE/rand	30	6.1809e-10	3.2125e-11
	60	7.0263e-13	1.7141e-13
DE/best	30	7.1967e-04	0
	60	3.9441e-04	0
MFDE	30	6.2083e-12	1.7308e-13
	60	3.8302e-17	0

Table 9
Experimental result of Rastrigrin

Function (f1)	Dim.	Avg.	Best
DE/rand	30	3.1961e+00	5.7124e-01
	60	1.0205e+02	7.9994e+01
DE/best	30	5.1738e-01	5.8664e-08
	60	3.7062e+01	1.9934e+00
MFDE	30	6.4222e-04	4.7275e-05
	60	3.1218e-02	8.7881e-04

All algorithms have no significantly difference in Rosenbrock function, epically in 60-dimension. Fig. 5 and 6 represents the convergence progress of function Rosenbrock.

MFDE has superority performance in high dimension and multi-modality function, Fig 7 to 10 represents the convergence progress of Griewank and Rastrigrin.Although DE/best provides powerful solving ability in single-modality but in multi-modality has premature convergence.

But MFDE can balance the performance between global exploration and local exploitation in high dimension and complex function.

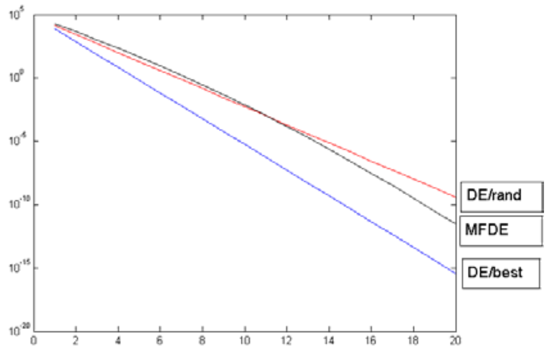


Fig. 3 Convergence progress of Sphere in 30 dimension

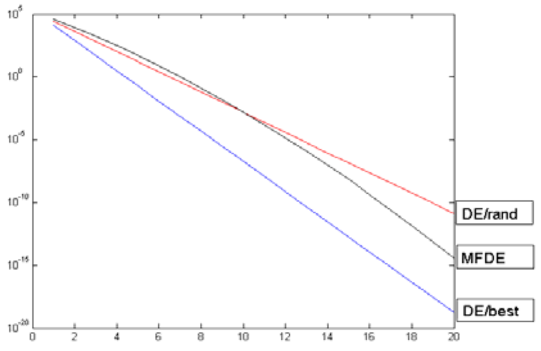


Fig. 4 Convergence progress of Sphere in 60 dimension

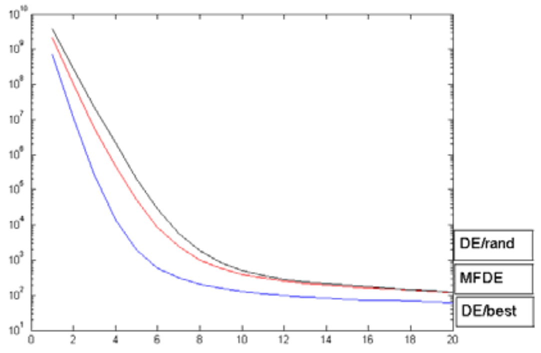


Fig. 5 Convergence progress of Rosenbrock in 30 dimension

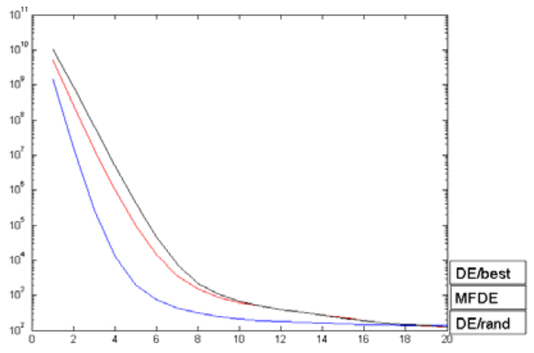


Fig. 6 Convergence progress of Rosenbrock in 60 dimension

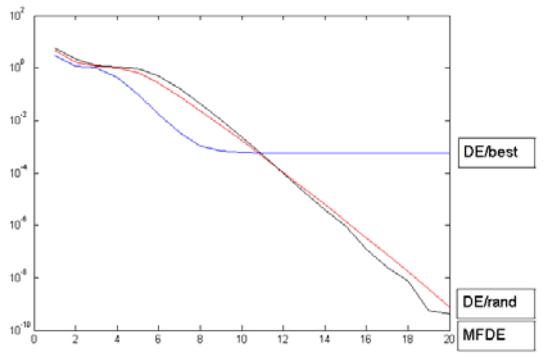


Fig. 6 Convergence progress of Griewank in 30 dimension

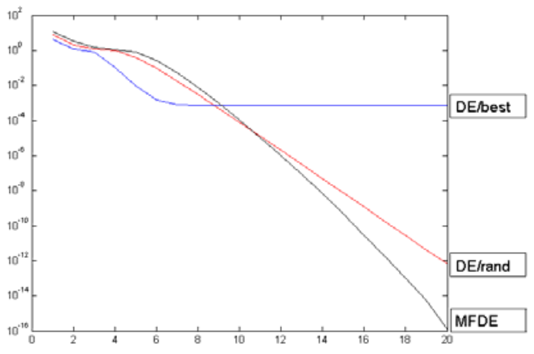
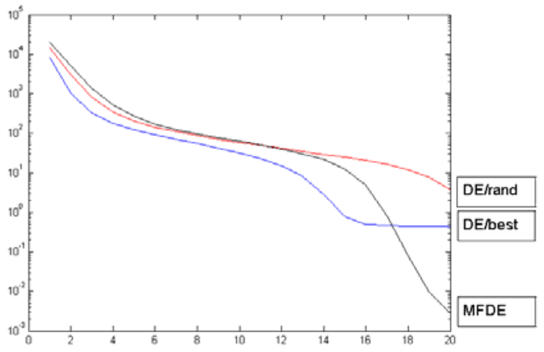


Fig. 7 Convergence progress of Griewank in 60 dimension



5.

Fig. 6 Convergence progress of Rastrigian in 30 dimension

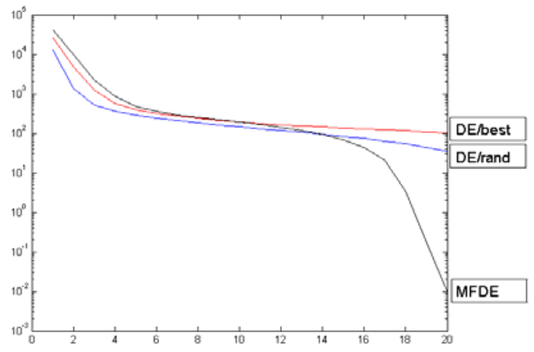


Fig. 7 Convergence progress of Rastrigian in 60 dimension

In this study, the proposed dynamic generate function is a mechanism that dynamical improve the mutation factor during the generation to promote overall performance. Compare with traditional DE version increasing the convergence rate, high exploration ability, lower probability to drop in region optimum. Experiment result represents the MFDE's performance is not significant in single-modality function but compare with DE/rand which we improve has significant improvement. MFDE has significant better outcome on multi-modality function than single-modality function. In another word, MFDE can bring outstanding performance in high complexity function. Because of DE/best characteristic fast convergence, in single-modality function has better performance instead high convergence to cause easy to drop in region optimum in multi-modality function cannot break away, therefore the efficiency of solving performance relative weak.

Besides this novel algorithm has low capacity of computation, under do not increasing the cost cause computation time to improve the performance. In the future, the mechanism "dynamic generate function" can import to any version of differential evolution without increase the time complexity.

6. References

[1] K. V. Price, "Differential evolution: a fast and simple numerical optimizer," Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American, 1996, pp. 524-527.W.-K. Chen, Linear Networks and Systems (Book style). *Belmont, CA: Wadsworth,* 1993, pp. 123–135

[2] R. Storn, "On the usage of differential evolution for function optimization," Fuzzy Information Processing Society, 1996. *NAFIPS. 1996 Biennial Conference of the North American*, 1996, pp. 519-523.

[3] Ali, M. M. and Torn, A. "Population set-based global optimization algorithms: some modications and numerical studies." *Comput. Oper. Res.*, vol.31, issue 10, 2004, pp. 1703-1725

[4] R. Storn, "Differential evolution research – trends and open questions," *Advances in Differential Evolution,* Volume 143, Heidelberg : Springer Berlin ,2008 ,pp. 1-31

[5] M. Rickert, O. Brock and A. Knoll, "Balancing exploration and exploitation in motion planning," Robotics and Automation, 2008. ICRA 2008. *IEEE International Conference on, 2008*, pp. 2812-2817

[6] Storn, R., Price, K., "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization* vol. 11, 1997, pp. 41–359

[7] K. Price, "An introduction to differential evolution", in: D.Corne, M. Dorigo, F. Glover (Eds.), *New Ideas in Optimization, McGraw-Hill, London*, 1999, pp. 79–108

[8] P. Kaelo and M. M. Ali, "A numerical study of some modified differential evolution algorithms," European *Journal of Operational Research*, vol. 169, 2006, pp. 1176-1184

[9] M.M. Ali, A. Törn, "Topographical differential evolution using pre-calculated differentials", in: G. Dzemyda, V. Saltenis, A. Zilinskas (Eds.), *Stochastic and Global Optimization, Kluwer Academic Publisher,* London, 2002, pp. 1–17