Improved Differential Evolution Based on Mutation Strategies

John Saveca[1], Zenghui Wang[1(✉)], and Yanxia Sun[2]

[1] Department of Electrical and Mining Engineering, University of South Africa, Johannesburg 1710, South Africa
Johnsaveca80@gmail.com, wangzengh@gmail.com

[2] Department of Electrical and Electronic Engineering Science, University of Johannesburg, Johannesburg 2092, South Africa
sunyanxia@gmail.com

Abstract. Differential Evolution (DE) has been regarded as one of the excellent optimization algorithm in the science, computing and engineering field since its introduction by Storm and Price in 1995. Robustness, simplicity and easiness to implement are the key factors for DE's success in optimization of engineering problems. However, DE experiences convergence and stagnation problems. This paper focuses on DE convergence speed improvement based on introduction of newly developed mutation schemes strategies with reference to DE/rand/1 on account and tuning of control parameters. Simulations are conducted using benchmark functions such as Rastrigin, Ackley and Sphere, Griewank and Schwefel function. The results are tabled in order to compare the improved DE with the traditional DE.

Keywords: Differential Evolution · Convergence speed · Mutation scheme Control parameters

1 Introduction

Differential Evolution (DE) has received much attention from various researchers and research institutions since its inception by Storn and Price two decades ago. DE's recognition involves its robustness, simplicity, speed and reliability to convergence to true optimum when solving an optimization problem. DE has gained much more success in series of benchmark academic competitions, black box global optimization competitions and real world optimization applications, leading to a big interest from both researchers and practitioners [6, 8]. Like other Evolutionary Algorithms (EA), DE uses a population based stochastic search method instead of complex mathematical operation [5]. By characteristics, DE is identified as an efficient and reliable global optimizer for different optimization fields such as constrained and unconstrained optimization, multimodal optimization and multi-objective optimization [6]. Despite DE algorithm being regarded as one of the best reliable and efficient EA method for solving optimization problems, it also has its own limitations. DE experiences stagnation, which in return deteriorates its performance. The occurrence of stagnation causes the algorithm not to get better solutions from the candidate solutions that are newly created, even though the diversity of the population remains [5]. Chances of stagnation occurrence depend on the availability of number of different potential trial vectors and their survival chances in the following generations [5]. In this paper, DE improvement is proposed, based on modification of mutation schemes and tuning of control parameters. The research will look to improve DE's convergence speed without experiencing stagnation. The improved DE will be used to optimize power quality in smart. Unlike Genetic Algorithm optimizing process which is affected by crossover function, in DE, mutation function plays a significant role during optimization [7]. DE's general notation is denoted as DE/X/Y/Z, where X indicates the mutation vector, Y indicated the number of difference vectors used and Z indicates the exponential or binomial crossover scheme [7]. As reported by [8], Differential mutation contains two parts, selection of base vector and summing of the difference vectors. The rest of the paper is organized as follows: Sect. 2 discusses background of the classical DE, evolutionary functions and mutation function schemes. Section 3 discusses the Improved DE algorithm formation and formulas. Section 4 gives the methodology to be followed during the experiment of DE improvement. Section 5 gives the results of the experiment and the discussion of the results. Section 6 gives the conclusion of the research.

2 Differential Evolution

Differential Evolution (DE) algorithm is one of the stochastic population-based evolutionary optimization algorithm that forms random search and optimization procedures by following natural evolutionary principles. Its term DE is due to existence of a special type of difference vector, as explained in [1]. During optimization, DE preserves candidate solutions population and creates new candidate solutions by combination of existing candidate solutions according to their simple formulae. The best candidate solution with better fitness on the optimization problem is kept close by [2]. DE uses three evolutionary functions during problem optimization, being mutation function, crossover function and selection function. Mutation function randomly generates variations to existing individuals to present new information into the population. The functioning creates mutation vectors vi, g at each generation g, based on the population of the current parent $\{X1,i,0 = (x1,i,0,\ x2,i,0,\ x3,i,0,\ \ldots,\ xD,i,0) | i = 1, 2, 3, \ldots., N\}$

$$\text{DE/rand/1} \quad vi, = Xr0, + Fi\,(Xr1, - Xr2,) \tag{1}$$

$$\text{DE/current-to-rest/1}\ vi, = Xi, + Fi\,(Xbest, - Xi,) + Fi\,(Xr1, - Xr2,) \tag{2}$$

$$\text{DE/best/1} \quad vi, = Xbest, + Fi\,(Xr1, - Xr2,) \tag{3}$$

where,

$r0, r1, r2$ = different integers uniformly chosen from the set $\{1, 2, \ldots.., N\}\backslash\{i\}$,
$Xr1, - Xr2,$ = different vector to mutate the parent,
$Xbest,$ = best vector at the current generation,
Fi = mutation factor which ranges on the interval (0, 1+).

The crossover function performs an exchange of information between different individuals in the current population. The final trial vector is formed by binomial crossover operation.

$$u_{i,g} = \left(u_{1,i,g}, u_{2,i,g}, \ldots, u_{D,i,g}\right) \tag{4}$$

$$u_{j_{,,}} = \begin{cases} vj, i, g\ldots\ldots \; if \; randj\,(0,1) \leq CRi \; or \; j = jrand \\ xj, i, g\ldots\ldots\ldots \; otherwise \end{cases} \tag{5}$$

where,

randj (*a*, *b*) = uniform random number on the interval (a, b) and newly generated for each *j*,
jrand = *jrandiant* (1, D) = integer randomly chosen from 1 to D and newly generated for each *i*.

The selection function passes a driving force towards the most favorable point by preferring individuals of better fitness. The selection operation selects the better one from the parent vector *Xi*, and the trial vector *ui*, according to their fitness values $f(\cdot)$ [3].

$$X_{i,+1} = \begin{cases} ui, g\ldots\ldots \; if \; f\,(ui, g) < Xi, g \\ Xi, g\ldots\ldots\ldots\ldots\ldots \; otherwise \end{cases} \tag{6}$$

Unlike other evolutionary algorithms, DE requires selection of only three control parameters, namely, Population Size (PS), Mutation Factor (F) and Crossover rate (Cr). According to [4], number of iterations ($\mathrm{Iter}_{\max}$) is not considered a control parameter, since some stopping criteria is need on the simulation. However, it is very helpful to have an estimation number of iterations in order to prevent a very long running time of the program. Mutation factor F value that can be selected ranges from 0.1 to 2.0 while the Crossover rate value ranges from 0.1 to 1.0. Population size is determined by the Dimensionality D of the objective function, where the values from 5D to 10D are suggested. However, the values are extended from 2D up to 40D [4].

3 Improved Differential Evolution

For DE improvement, two factors have been considered, the first being tuning of control parameters to get the suitable combination to be used on a selected mutation scheme. In this case the selected mutation scheme is DE/rand/1. DE/rand/1 is the most commonly used scheme due to its simplicity and fast convergence during optimization of the problem. The second factor is modification of selected mutation scheme. Three modifications schemes are developed by taking Mutation Factor F into account on mutation formula. As reported by [9], it is mutation that separates one DE strategy from another. Mutation is responsible for expansion and exploration of the search space in order to obtain the optimum solution for a given optimization problem, by combining different parameter vectors in such a way that a new population vector, termed donor vector is generated [1]. F is responsible for the amplification of the differential variation [1–10]. In this modification, F will also be used to amplify the base vector $Xr0$ in order to explore much wider search space for better optimum solution. For the first modification, the individual vector $Xr0$ is squired and divided by mutation factor F as shown in the formula. The formula will be named DE/Modi/1

$$\mathrm{DE/Modi/1} \quad vi, = (Xr0)^2, \div \mathrm{F}i + Fi\,(Xr1, -Xr2,) \tag{7}$$

where,

$r0$, $r1$, $r2$ = different integers uniformly chosen from the set $\{1, 2, \ldots .., N\}\backslash\{i\}$,
$Xr1$, $- Xr2$, = different vector to mutate the parent,
$Xr0$, = base vector,
Fi = mutation factor which ranges on the interval (0, 1+).

On the second modification, the parent vector is multiplied by the mutation factor F. In this case the individual vector is not squired as shown in the next formula. The formula will be named DE/Modi/2

$$\mathrm{DE/Modi/2} \quad vi, = \mathrm{F}i \times Xr0,\; + Fi\,(Xr1, -Xr2,) \tag{8}$$

The third and final modification involves three factors applied to the individual vector. First it is squired as done on the first modification, secondly it is multiplied by the mutation factor, thirdly, it is divided by 2 as shown in the formula. The formula will be named DE/Modi/3

$$\mathrm{DE/Modi/3} \quad vi, = \mathrm{F}i \times (Xr0)^2 \div 2 + Fi\,(Xr1, -Xr2,) \tag{9}$$

4 Methodology

The following steps, which are also shown in Fig. 1, were taken during simulation of the DE improvement by means of control parameter tuning and mutation scheme modification. The following benchmark functions were used to during simulation of the experiment, Rastrigin function, Ackley function, Sphere Function, Griewank Function. Schwefel Function, Bukin Function. SumPower Function and SumSquare Function. DE/rand/1 is selected for the experiment.

Step 1: DE/rand/1 pseudo-code is done on matlab and Setting of control parameters is done in the following manner, the constant parameters: D = 2, PS = 50, I_-max = 200. The varying parameters: F = [0.1–2.0], Cr = [0.1–1.0].
Step 2: Each benchmark function mentioned above is tested by varying F and C from 0.1 to 2.0 and from 0.1 to 1.0 respectively in order to determine the perfect set of values that makes a fast convergence on the optimization process.
Step 3: The determined set values of F and C are then used in the three modified mutation schemes without being varied. In this case the determined set Combination values of F/C are 0.2/0.2, 0.2/0.3, 0.2/0.5, 0.2/0.7, 0.2/0.9, 0.1/0.9, 0.4/0.9, 0.6/0.9

and 0.8/0.9. The results of all the convergence of all the above mentioned benchmark functions during F/C combination are tabled and will be compared with the results of convergence that are obtained on the modified mutation schemes.

Step 4: The original mutation scheme DE/rand/1 formula is modified according to the above mentioned mutation schemes modifications. The control parameters on the modified mutation schemes are, F = 0.2, 0.1, 0.4, 0.6 and 0.8, C = 0.9, 0.2, 0.3, 0.5, 0.7 and 0.9, D = 2, PS = 50, I_max = 200.

Step 5: All the benchmark functions are simulated for convergence speed for DE/Modi/1, DE/Modi/2 and DE/Modi/3. The results are tabled and compared with the results of DE/rand/1.

Step 6: Statistical data and time complexity will be determined.

Following is the pseudo-code for DE with one of the modified mutation scheme DE/Modi/3.

```
Set NP, C, F, parameters
initialize population p= {$x_1$, $x_2$, $x_3$...$x_m$}, $x_1 \in D$
repeat
for i=1 to NP do
        Generate MxN matrix
        for m=1:M
          for n=1:N
             X(m,n)=X_min(n)+rand()*(X_max(n)-X_min(n));
          end
        end
    generate a new mutant vector
    $y = (x_{r1})^2 * F_x / 2 + F_x * (x_{r2} - X_{r3})$
                if f(y) < f(x) then inert y into the new generation
                   else insert x into new generation
                end
   end
until stopping criteria
```

Fig. 1. DE/Modi/3 pseudo-code.

5 Results and Discussion

Follows are the results obtained during simulation of the five benchmark functions. All simulations are ran up maximum of 200 iterations. All simulations are run according to the following parameters: I_max = 200 iterations, D = 2, PS = 50 and F/C combination = 0.2/0.2, 0.2/0.3, 0.2/0.5, 0.2/0.7, 0.2/0.9, 0.1/0.9, 0.4/0.9, 0.6/0.9 and 0.8/0.9.

Following are the results of DE/rand/1 strategy as shown in Figs. 2, 3, 4, 5, 6 and Table 1.

From the above results of DE/rand/1, it can be noticed that most functions convergence becomes more strong after 63 iterations. For Griewank function, the

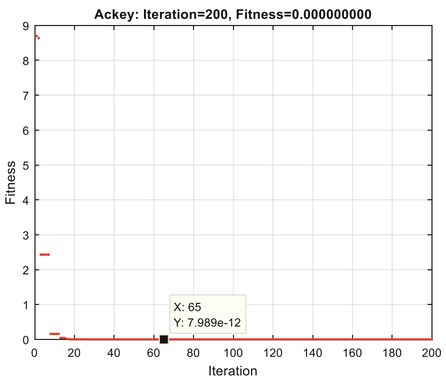

Fig. 2. Ackley's fitness vs iterations using DE/rand/1.

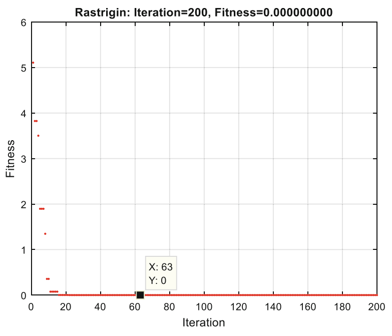

Fig. 3. Rastrigin's fitness vs iterations using DE/rand/1

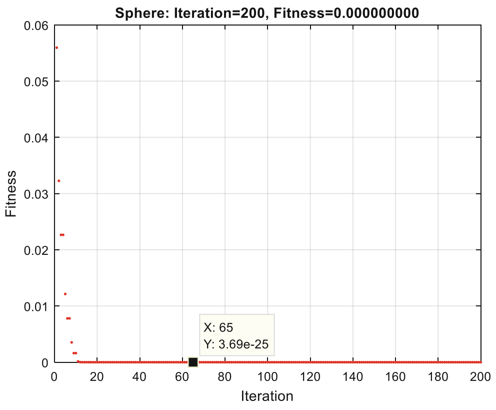

Fig. 4. Sphere's fitness vs iterations using DE/rand/1.

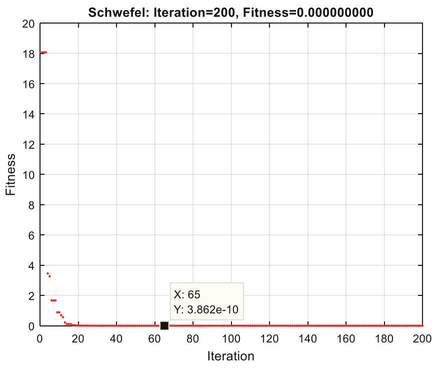

Fig. 5. Schwefel's fitness vs iterations using DE/rand/1.

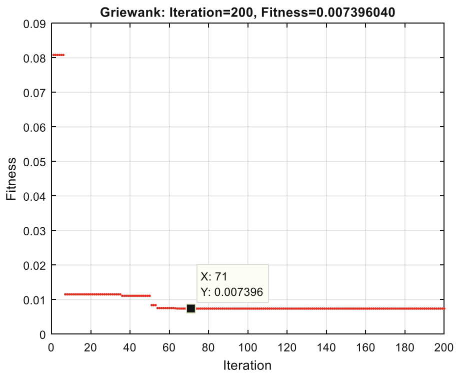

Fig. 6. Griewank's fitness vs iterations using DE/rand/1.

Table 1. DE/rand/1 results

Benchmark function	DE/rand/1	
	Iterations	Fitness
Ackley	65	7.989e−12
Rastrigin	63	0
Sphere	65	3.69e−25
Schwefel	65	3.862e−10
Griewank	71	0.007396

convergence is not much strong as it generates the fitness of 0.007396 after 71 iterations. Therefore it means Griewank function will require more generations in order for it to have a strong convergence of fitness zero (0) or close to zero (0).

Follows are the results of DE/Modi/1 strategy as shown in Figs. 7, 8, 9, 10, 11 and Table 2.

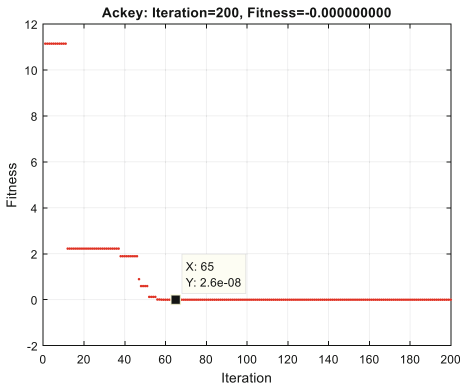

Fig. 7. Ackley's fitness vs iterations using DE/Modi/1.

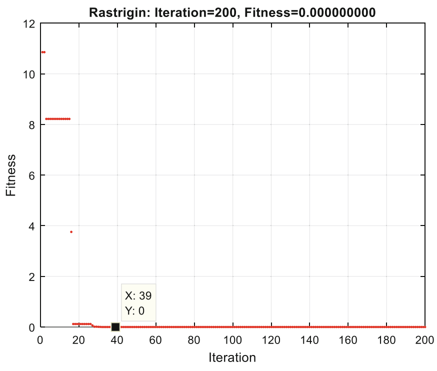

Fig. 8. Rastrigin's fitness vs iterations using DE/Modi/1

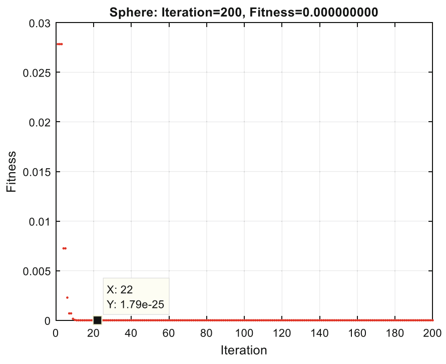

Fig. 9. Sphere's fitness vs iterations using DE/Modi/1.

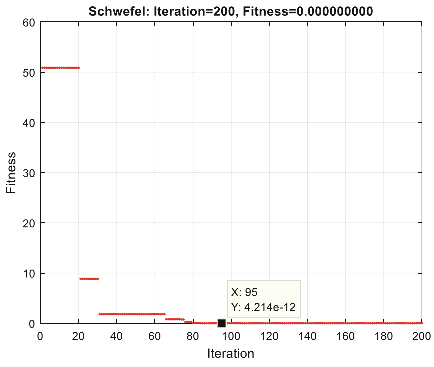

Fig. 10. Schwefel's fitness vs iterations using DE/Modi/1.

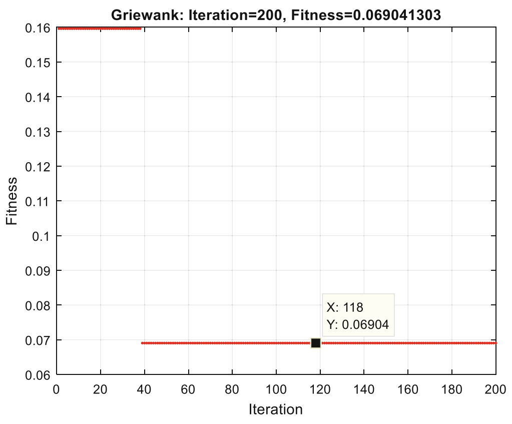

Fig. 11. Griewank's fitness vs iterations using DE/Modi/1.

Table 2. DE/Modi/1 results

Benchmark function	DE/Modi/1	
	Iterations	Fitness
Ackley	65	2.6e−08
Rastrigin	39	0
Sphere	22	1.79e−25
Schwefel	95	4.214e−12
Griewank	118	0.06904

From the above results of DE/Modi/1, it can be noticed that the convergence of the strategy is slightly slow for Ackley function. For Griewank, the convergence turns to be very weak and slow compared to classical DE strategy. For Schwefel function the convergence is robust but extremely slow compared to DE/Rand/1 strategy. The convergence speed improved for Rastrigin and Sphere functions.

Table 3 is the results of DE/Modi/2 strategy.

Table 3. DE/Modi/2 results

Benchmark function	DE/Modi/2	
	Iterations	Fitness
Ackley	17	1.863e−11
Rastrigin	14	0
Sphere	19	1.04e−28
Schwefel	18	6.02e−11
Griewank	21	0

From the results obtained from DE/Modi/2, it can be noticed that convergence speed robustness of all the functions has improved compared to both above strategies DE/Rand/1 and DE/Modi/1. In this strategy, Griewank function is able to reach a robust convergence.

Table 4 is the results obtained from DE/Modi/3 strategy.

Table 4. DE/Modi/3 results

Benchmark function	DE/Modi/3	
	Iterations	Fitness
Ackley	16	2.538e−11
Rastrigin	12	0
Sphere	15	1.189−26
Schwefel	24	4.09e−13
Griewank	29	0

From the results of DE/Modi/3, it can be noticed that there is a slight change of convergence speed between DE/Modi/2 and DE/Modi/3. Convergence speed for Ackley and Rastrigin function slightly improved compared to DE/modi/2, while for Sphere, Schwefel and Griewank function, convergence speed slightly dropped compared to DE/Modi/2.

6 Conclusion

Based on the above results, it has been noticed that DE/Modi/1 results lack robustness and convergence speed, making just 41.67% of best convergence time. For DE/Modi/3 it can be noticed that improvement is achieved compared to classical DE/Rand/1 with a percentage of 81.94% of the best convergence time. DE/Modi/2 achieved the best results with 84.72% of best convergence time compared to all other modified strategies on this paper with best convergence speed and strong convergence. On the other hand F/C combination of 0.1/0.9 produced fast and robust convergence during the DE/Modi2 and DE/Modi3 simulation session, making it the best Mutation Factor/Crossover Rate combination for the DE/Modi2 and DE/Modi3 mutation strategies. It can be concluded that DE convergence speed has been improved through modified strategies DE/Modi2 and DE/Modi/3 with F/C combination of 0.1/0.9. The modified mutation strategies DE/Modi2 and DE/Modi/3 with F/C combination of 0.1/0.9 can be used in future due to their robust convergence, fast and effective convergence speed and ability to optimize other functions that classical DE/rand/1 is not able to optimize to minimum optimal point such as Griewank and Bukin function.

Acknowledgments. This research is supported partially by South African National Research Foundation Grants (No. 112108 and 112142), and South African National Research Foundation Incentive Grant (No. 95687), Eskom Tertiary Education Support Programme Grants (Z. Wang, Y. Sun), Research grant from URC of University of Johannesburg.

References

1. Chattopadhyay, S., Sanyal, S.K., Chandra, A.: Comparison of various mutation schemes of differential evolution algorithm for the design of low-pass FIR filter, pp. 809–814 (2011)
2. Sagoo, S.: Array failure correction using different optimization techniques, MTech thesis (2016)
3. Ganbavale, M.P.: Differential evolution using matlab. Birla Institute of Technology and Science, Pilani, Hyderabad Campus (2014)
4. Penunuri, F., Cab, C., Tapia, J.A., Zambrano-Arjona, M.A.: A study of the classical differential evolution control parameters. Swarm Evol. Comput. **26**, 86–96 (2015)
5. Zheng, L.M., Zhang, S.X., Tang, K.T., Zheng, S.Y.: Differential evolution powered by collective information. Inf. Sci. **399**, 13–29 (2017)
6. Wu, G., Shen, X., Chen, H., Lin, A., Suganthan, P.N.: Ensemble of differential evolution variants. Inf. Sci. **423**, 172–186 (2017)
7. Thangaraj, R., Pant, M., Abraham, A.: New mutation schemes for differential evolution algorithm and their application to the optimization of directional over-current relay settings. Appl. Math. Comput. **216**, 532–544 (2010)
8. Opara, K., Arabas, J.: Comparizon of mutation strategies in differential evolution-a probabilistic perspective. Swarm Evol. Comput. **338**, 1–37 (2017)
9. Tayal, D., Gupta, C.: A new scaling factor for differential evolution optimization. In: National Conference on Communication Technologies & Its Impact on Next Generation Computing CTNGC2012 Proceedings, IJCA, pp. 1–5 (2012)
10. Sarker, R.A., Elsayed, S.M., Ray, T.: Differential evolution dynamic parameters section for optimization problems. IEEE Trans. Evol. Comput. **18**, 689–707 (2014)