RDE - Reconstructed Mutation Strategy for Differential Evolution Algorithm

Meera Ramadas[1(✉)], Ajith Abraham[2], and Sushil Kumar[3]

[1] AIIT, Amity University, Noida, India
meera_mgr@rediffmail.com
[2] MIR Labs, Auburn, USA
[3] ASET, Amity University, Noida, India

Abstract. Several researchers' innovative work during past years has led to development of numerous optimization techniques. Complex task that were once difficult to be compute using traditional methods now can use the optimization techniques for computation. Differential Evolution (DE) is a powerful, population based, stochastic optimization algorithm. The mutation strategy of DE algorithm is an important operator as it aids in generating a new solution vector. In this paper, we are introducing a variant of DE mutation strategy named RDE (Reconstructed Differential Evolution). This strategy use three different control parameters. The results computed here are then compared with the results of an existing mutation strategy where in the comparison show a better performance for the new revised strategy.

Keywords: Optimization · Mutation · Control parameters · Differential evolution

1 Introduction

The idea of natural selection and biological evolution propounded by Charles Darwin resulted in the concept of Evolutionary algorithm (EA). For computing a particular problem, an environment is created where in the potential solution can evolve. Environment is framed by the guidelines of the problem and fortifies the evolution of good solution. EA is a good technique for searching the optimal solutions. EA uses various concepts of reproduction, selection, mutation and recombination. Various evolutionary algorithms have been developed in due course of time. Initially researchers used the genetic algorithm technique for solving complex problems. In 1997, Storn and Price proposed the Differential Evolution (DE) technique. Like various other evolutionary algorithms, DE is also a population based stochastic method. DE is one of the best evolutionary algorithm for solving real valued test functions.

Numerous attempts are being done to improve the performance of DE for several specific applications. The efficiency and performance of DE greatly depends on the trial vector generation strategy and the control parameters used. Several variants of the existing technique are developed by changing these trial vector strategy and control

A. Abraham et al. (eds.), *Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016)*, Advances in Intelligent Systems and Computing 614, DOI 10.1007/978-3-319-60618-7_8

parameters. The three control parameters in DE algorithm are: mutation scale factor F, crossover constant and the population size.

In this paper, a variant of the mutation strategy named RDE is proposed by using three different mutation scale factors. One of the scale factor is a constant value and the other two scale factors are variable values between the range (0,1) where one factor is the complement of the other value. This strategy showed better efficiency compared to the existing mutation strategies.

2 Background Study

Das et al. [6] proposed two new variants of DE, DE with random scale factor (DERSF) and DE with time varying scale factor(DETVSF). The new method showed statistically improved results. Brest et al. [5] presented a new version of DE with self-adaptive control parameter settings showing better efficiency in comparison to the existing techniques. Grosan et al. [9] gave the need for hybridizing evolutionary algorithms and proposed the possibilities on hybridization. Also a review on the existing hybrid techniques were also stated. Das et al. [8] gave a detailed study on particle swarm optimization (PSO) and DE. Subsequently, a mutual synergy of PSO and DE were discussed and results computed. Ali et al. [1] proposed the technique of using nonlinear simplex method with pseudo number to generate the initial population. The method was named as NSDE and results were tabulated and compared. Xin et al. [10] developed a novel adaptive hybrid of PSO and DE (HPSO-DE). This technique maintains the diversity of population.

Gong et al. [3] presented a set of improved DE that try to adaptively choose a suitable strategy for a problem at hand. In this paper, different parameter adaption methods of DE are used for different strategies. The efficiency of the technique was tested and verified. Islam et al. [2] proposed a new mutation strategy using fitness induced parent selection for binomial crossover of DE and a scheme of adapting two of the control parameters to achieve better results. Gong et al. [4] proposed ranking based mutation operator for DE algorithm. Here the selection of the parent in mutation operation is selected according to the ranking in the current population. The proposed operators were integrated into advanced DE variants to verify its effects. The proposed mutation operators enhanced the performance of DE algorithm.

Yu et al. [14] proposed an adaptive DE (ADE) algorithm with new mutation strategy and a two level adaptive parameter control scheme. This technique has a good balance between population diversity and fast convergence. Tang et al. [11] developed a novel variant of DE with an individual dependent mechanism that uses an individual dependent parameter (IDP) and an individual dependent mutation (IDM) strategy. Tabulated results show greater performance to classical technique. Qiu et al. [12] developed the simultaneous use of individuals across generations from objective based perspective. Results obtained show the statistical superiority of the proposed technique to several evolutionary algorithms. Ramadas et al. [9] proposed an algorithm ssFPA/DE where Differential evolution approach was combined with the concept of Flower Pollination Algorithm. The proposed technique gave better results in comparison to tradition DE

approach. Ramadas et al. [10] also proposed an new mutation strategy named ReDE – a revised mutation strategy. This strategy used two control parameters and two types of population. The efficiency of the new technique was better than the traditional approach.

3 Differential Evolution

In a search space of n-dimensions of likely solutions, a specified number of vectors are arbitrarily identified. In each iteration or generation, a new vector will be formed by combining two or more vectors which are arbitrarily identified from the population. The outcome vector is with predetermined target vector. A trial vector is created in a process called recombination. If it produce a better value of objective function, then the trial vectors are accepted in next generation. Until some stopping criteria is satisfied, the mutation, recombination and selection are continued. DE use the population of NP candidate solutions denoted as $X_{i,G}$ where $i = 1, 2 \ldots$NP where index i denote population and G represents generation of population. Differential Evolution algorithm depends on the three operations mainly mutation, selection and reproduction.

Mutation: This operator causes DE to be distinct from other Evolutionary algorithms. It computes the weighted difference between the vectors in population. Mutation starts by arbitrarily choosing three individuals from the population. This operation extends the workspace. For a given parameter $X_{i,G}$ we are arbitrarily selecting 3 vectors $X_{r1,G}, X_{r2,G}$ and $X_{r3,G}$ such that r_1, r_2, r_3 are distinct. Then the donor vector $V_{i,G}$ is computed as:

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

Here F is the mutation factor which is a constant from [0,1]. The above strategy is denoted as DE/rand/1. Mutation function demarcates one DE scheme from another. The often used DE codes are given below:

$$\text{DE/rand/2} \quad V_{i,G} = X_{r1,G} + F \times (X_{r2,G} - X_{r3,G}) + F \times (X_{r4,G} - X_{r5,G}) \tag{2}$$

$$\text{DE/best/1} \quad V_{i,G} = X_{best,G} + F \times (X_{r1,G} - X_{r2,G}) \tag{3}$$

$$\text{DE/best/2} \quad V_{i,G} = X_{best,G} + F \times (X_{r1,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G}) \tag{4}$$

$$\text{DE/rand} - \text{to} - \text{best/1} \quad V_{i,G} = X_{r1,G} + F \times (X_{best,G} - X_{r2,G}) + F \times (X_{r3,G} - X_{r4,G}) \tag{5}$$

where, $i = 1 \ldots NP$, $r_1, r_2, r_3 \in \{1, \ldots, NP\}$ are randomly selected and satisfy: $r_1 \neq r_2 \neq r_3 \neq i$, $F \in [0, 1]$, F is the control parameter proposed by Storn and Price.

Crossover: This process also termed as recombination, includes successful solutions into the population. The trial vector $U_{i,G}$ is created for target vector $X_{i,G}$ through binomial crossover. Components of donor vector enter trial vector with probability $C_r \in [0, 1]$. C_r is the crossover probability which is selected along with population size $NP \geq 4$.

$$U_{j,i,G+1} = \begin{cases} V_{j,i,G+1} & if\ rand_{i,j}[0,1] \leq C_r or\ if\ j = I_{rand} \\ X_{j,i,G+1} & if\ rand_{i,j}[0,1] > C_r or\ if\ j \neq I_{rand} \end{cases} \tag{6}$$

Here $rand_{i,j} \approx \cup[0, 1]$ and I_{rand} is random integer from 1,2…N.

Selection: This operation differs from the selection operation of other evolutionary algorithms. Here the population for next generation is chosen from vectors in current population and its subsequent trial vectors. The target vector $X_{i,G}$ is matched with the trial vector $V_{i,G}$ and the least value of function is taken into next generation.

$$X_{i,G+1} \begin{cases} U_{i,G+1} & if\ f(U_{i,G+1}) \leq f(X_{i,G})\ where\ i = 1, 2, \ldots N \\ X_{i,G} & otherwise \end{cases} \tag{7}$$

4 Reconstructed Mutation Strategy

In RDE, we have used three control parameters. By involving the best solution vector, this strategy coincides faster as compared to the traditional strategies having random vectors only. The variables $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$ are chosen at random. The parameter F known as amplifying parameter takes a constant value. The new parameter N1 takes a varying value which lies between (0,1) and N2 takes the complement of N1. As we are taking three different control parameters, the value of donor vector is improved greatly and hence the efficiency of DE algorithm is enhanced immensely. The proposed strategy is given as:

$$X' = X_{r1,G} + F * \left(N1 * \left(X_{best,G} - X_{r2,G}\right) - N2 * \left(X_{best,G} - X_{r3,G}\right)\right) \tag{8}$$

5 Experimental Settings

The above stated variant was implemented using MATLABr2008b on i7 core processor, 64 bit operating system with 12 GB RAM. A comparative result was obtained with the traditional mutation strategies. Here, we have taken 5 traditional mutation strategy (DE/rand/, DE/rand/2, DE/best/1, DE/best/2, DE/rand-to-best/1) and the proposed technique RDE and values obtained were compared. The traditional mutation strategies were replaced with the proposed mutation strategy and RDE was composed. In the experiment conducted, mutation constant F is given the value 0.6 and the crossover probability C_r is given the value 0.8. We have taken fifteen different functions and calculated the results by fixing the value to reach and number of iterations. We have also tested the strategy

by fixing the dimension as 25. One of the results obtained and their corresponding graphs are given below:

Based on Best Value after 50 runs (vtr = 1.e − 015):

Table 1. Best value after 50 runs for different functions

Function	DE					RDE
	DE/best/1	DE/rand/1	DE/best-to-rand/1	De/best/2	DE/rand/2	
Sphere	9.73e − 016	6.9e − 016	7.532e − 016	9.655e − 016	7.17e + 0	**6.6e − 016**
Beale	3.265e − 016	**2.318e − 016**	3.713e − 016	7.587e − 016	7.725e − 016	7.7e − 016
Booth	3.497e − 016	2.0514e − 016	6.0738e − 016	7.0792e − 016	8.35e − 016	**9.2e − 015**
Schwefel	**− 1.8e + 003**	**− 2.253e + 003**	**−7.8403e + 001**	**− 1.38e +003**	**−1.66e +003**	**−3.3e + 002**
Michlewicz	**− 7.6399e +00**	**− 7.214e + 00**	**−7.39e + 00**	**− 6.959 e + 00**	**−6.847 e + 00**	**−1.14e + 001**
Schaffer N.2	6.6e − 016	8.88e − 016	4.43e − 016	6.55e − 016	8.87e − 016	6.66e − 016
Schaffer N.4	3.05e − 015	2.9e − 001	2.92 − 001	2.93e − 001	2.89e − 001	**2.85e − 001**
HimmelBlau	1.6e − 016	8.05e − 016	3.83e − 016	9.12e − 016	**1.46e − 016**	4.38e − 016
Bird	**− 1.035e + 002**	**− 1.067e + 002**	**−1.05e + 002**	**− 1.065e + 002**	**− 1.03e + 002**	**− 1.05e + 002**
Extended cube	3.31e − 015	4.98e − 005	6.1e − 008	1.93e − 005	2.68e + 00	2.04e − 007
Ackeley	7.19e − 015	**6.46e − 012**	7.99e − 015	3.63e − 013	3.09e + 00	4.4e − 015
Gold	3.00e + 00	3.00e + 00	3.00e + 00	3.00e + 00	3.00e + 00	3.0e + 00
Griewank	9.99e − 016	9.99e − 016	1.6e − 013	6.56e − 013	1.07e + 00	9.99e − 016
Rastrigin	1.79e + 001	1.23e + 002	7.47e + 001	1.28e + 002	1.52e + 002	8.04e + 001
Rosenbrock	9.6e − 016	**1.07e − 008**	7.88e − 016	3.9e − 009	1.07e + 005	7.48e − 016

Based on NFE on fixed VTR for size = 25 (VTR = 1.e − 015) (Table 2):

Table 2. Based on NFE on fixed VTR for different functions for size = 25 (VTR = 1.e − 15)

Function	DE					
	DE/best/1	DE/rand/1	DE/best-to-rand/1	De/best/2	DE/rand/2	RDE
Sphere	2880000	3705000	313000	3260000	5000000	**902000**
Beale	**48000**	94000	67000	85000	127000	68000
Booth	500000	90000	71000	77000	118000	**69000**
Schwefel	7000	12000	13000	**4000**	6000	11000
Michlewicz	1000	1000	1000	1000	1000	1000
Schaffer N.2	**68000**	148000	119000	139000	224000	99000
Schaffer N.4	5000000	5000000	5000000	5000000	5000000	5000000
HimmelBlau	**45000**	95000	67000	77000	199000	67000
Bird	1000	1000	1000	1000	1000	1000
Extended cube	5000000	5000000	5000000	5000000	5000000	5000000
Ackeley	5000000	5000000	5000000	5000000	5000000	5000000
Gold	5000000	5000000	5000000	5000000	5000000	5000000
Griewank	2880000	4579000	500000	500000	500000	**1688000**
Rastrigin	5000000	5000000	5000000	5000000	5000000	5000000
Rosenbrock	**60900**	5000000	77000	5000000	5000000	1919000

Based on elapsed time of CPU in seconds for size = 25 (VTR = 1.e − 015) (Table 3):

Table 3. Based on elapsed time of CPU in seconds for different functions for size = 25 (VTR = 1.e − 015)

Function	DE					
	DE/best/1	DE/rand/1	DE/best-to-rand/1	De/best/2	DE/rand/2	RDE
Sphere	37.670	139.169	**16.318**	122.399	225.799	57.711
Beale	48.163	9.5786	**8.263**	8.533	8.81	18.35
Booth	11.6	8.25	6.817	8.494	8.5762	20.85
Schwefel	17.712	5.337	**4.3044**	4.956	4.5789	14.45
Michlewicz	2.73	2.217	2.817	**2.064**	2.074	9.02
Schaffer N.2	36.311	23.22	**17.3**	18.86	21.19	23.27
Schaffer N.4	**190.07**	227.9	261.1	238.23	245.6	234.07
HimmelBlau	14.8	16.94	18.4	**12.3**	15.95	16.74
Bird	12.39	11.8	8.25	8.71	8.07	**7.7**
Extended cube	**186.78**	372.56	345.3	352.3	335.02	330.7
Ackeley	314.6	310.42	325.56	312.2	**300.4**	345.6
Gold	**257.45**	425.8	331.62	312.52	319.5	334.5
Griewank	321.7	305.3	346.58	342.76	341.01	**157.7**
Rastrigin	**254.69**	316.8	265.4	310.1	323.52	301.2
Rosenbrock	52.01	36.23	52.79	324.06	34.53	**34.4**

A comparative analysis was performed and study done on each of the technique. By setting the dimension as 25 and value-to-reach (VTR) as e − 015, the best value, number of function evaluation (NFE) and the CPU time of different function strategies were calculated. It was noted that the proposed hybrid algorithm gave the best value for most of the standard functions.

6 Graphical Results

The above tabulated values were represented in a graphical form. The graphs show performance curve of six different function strategies. The x-axis represents the number of function evaluation for each mutation strategy and y-axis represents the objective function. The graph is plotted for the various values at each iteration for fixed VTR value of e − 015 and dimension size of 25 (Fig. 1).

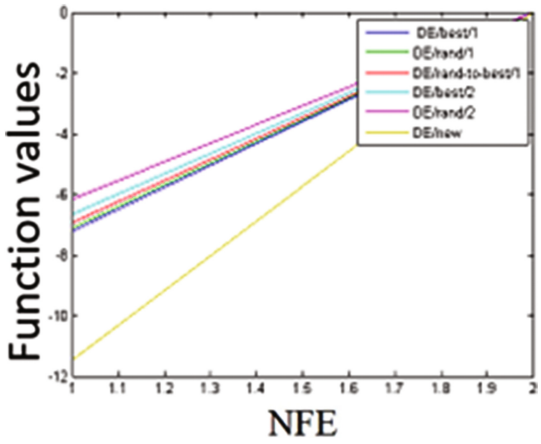

Fig. 1. Graphical representation of Michelawicz function

A comparative study was done based on above graphs. The study showed that the revised mutation strategy gave better results compared to the existing mutation strategy for various functions (Fig. 2).

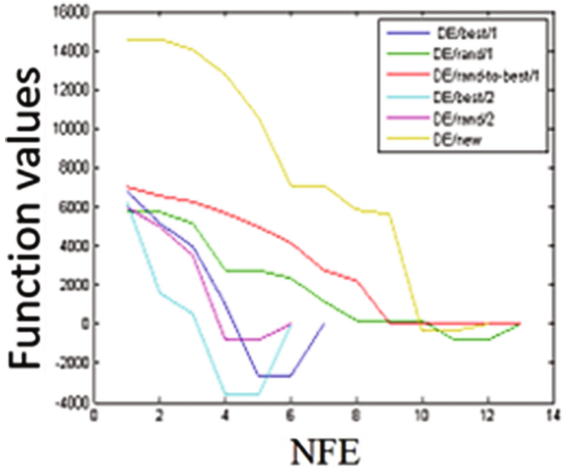

Fig. 2. Graphical representation for Schwefel function

7 Statistical Test

Friedman test was applied and the results obtained were tabulated on the values obtained from Table 1. Table 4 represents the values obtained from the test. N represents the population size and df represents the degree of freedom associated with the source. Asymptotic significance gives the p value of the Friedman test and Chi sq gives the

Friedman chi square statistics value. Table 5 depicts the rank position of the various mutation strategies used based on best value, NFE and CPU time (Fig. 3).

Table 4. Test statistics using Friedman's test

N	50
Chi sq	22.71
Df	5
Asymptotic Significance	0.0004

Table 5. Ranks of the different strategies

Strategies	Mean rank on best value	Mean rank on NFE	Mean rank on CPU time
DE/best/1	2.67	3.03	3.4
De/rand/1	3.23	4.3	3.7
DE/best-to-rand/1	2.60	3.0	3.3
De/best/2	4.23	3.4	3.2
DE/rand/2	5.1	4.2	3.1
New	3.16	3.1	4.1

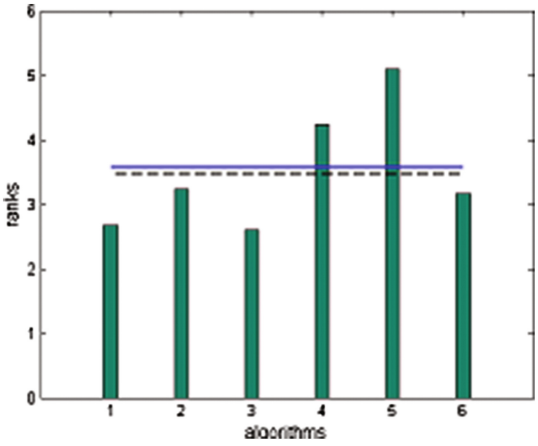

Fig. 3. Bonferroni Dunn chart for best value

The above tables show that the new mutation strategy has significant performance in comparison to the existing mutation strategies. Based on the ranks obtained, a graphical representation of the results is shown below. The x axis of the graph represents the six different mutation strategies used and the y axis shows the ranks obtained for each strategy based on different parameters like best value obtained and NFE value obtained (Fig. 4).

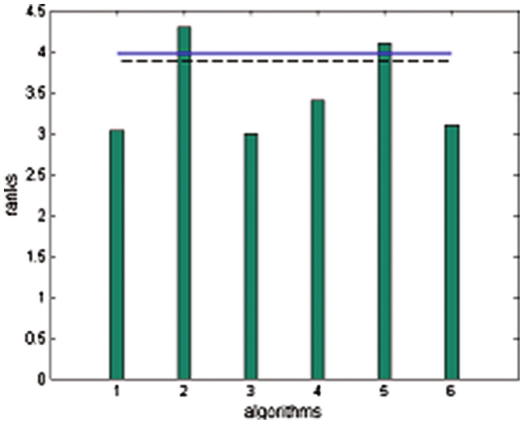

Fig. 4. Bonferroni Dunn bar chart for NFE

8 Conclusion

In this paper, we have given a simple, efficient mutation strategy. RDE strategy was compared against the existing mutation strategy. The comparative study showed that the proposed strategy gave much better for most of the functions evaluated. A detailed study was done and graphs were plotted. Further the work can be extended to the field of clustering for verifying the performance of the new mutation strategy in that area.

References

1. Ali, M., Pant, M., Abraham, A.: Simplex differential evolution. Acta Polytech. Hung. **6**(5), 95–115 (2009)
2. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006)
3. Das, S., Abraham, A., Konar, A.: Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In: Advances of Computational Intelligence in Industrial Systems, pp. 1–38. Springer, Heidelberg (2008)
4. Das, S., Konar, A., Chakraborty, U.K.: Two improved differential evolution schemes for faster global search. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation, pp. 991–998. ACM (2005)
5. Gong, W., Cai, Z.: Differential evolution with ranking-based mutation operators. IEEE Trans. Cybern. **43**(6), 2066–2081 (2013)
6. Gong, W., Cai, Z., Ling, C.X., Li, H.: Enhanced differential evolution with adaptive strategies for numerical optimization. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **41**(2), 397–413 (2011)
7. Grosan, C., Abraham, A.: Hybrid evolutionary algorithms: methodologies, architectures, and reviews. In: Hybrid Evolutionary Algorithms, pp. 1–17. Springer, Heidelberg (2007)

8. Islam, S.M., Das, S., Ghosh, S., Roy, S., Suganthan, P.N.: An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **42**(2), 482–500 (2012)
9. Ramadas, M., Pant, M., Abraham, A., Kumar, S.: ssFPA/DE: An efficient hybrid differential evolution–flower pollination algorithm based approach. Int. J. Syst. Assur. Eng. Manag. 1–14 (2016)
10. Ramadas, M., Abraham, A., Kumar, S.: ReDE- A revised mutation strategy for differential evolution algortihm. Int. J. Intell. Eng. Syst. **9**(4), 51–58 (2016)
11. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Trans. Evol. Comput. **12**(1), 107–125 (2008)
12. Qiu, X., Xu, J.-X., Tan, K.C., Abbass, H.A.: Adaptive cross-generation differential evolution operators for multiobjective optimization. IEEE Trans. Evol. Comput. **20**(2), 232–244 (2016)
13. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
14. Tang, L., Dong, Y., Liu, J.: Differential evolution with an individual-dependent mechanism. IEEE Trans. Evol. Comput. **19**(4), 560–574 (2015)
15. Xin, B., Chen, J., Peng, Z., Pan, F.: An adaptive hybrid optimizer based on particle swarm and differential evolution for global optimization. Sci. China Inf. Sci. **53**(5), 980–989 (2010)
16. Yu, W.J., Shen, M., Chen, W.N., Zhan, Z.H., Gong, Y.J., Lin, Y., Liu, O., Zhang, J.: Differential evolution with two-level parameter adaptation. IEEE Trans. Cybern. **44**(7), 1080–1099 (2014)