ICIC Express Letters
Volume **12**, Number **8**, August **2018**

 ISSN 1881-803X
pp. **785–790**

GROWING DIFFERENTIAL EVOLUTION METHOD USING RANKING INFORMATION OF SEARCH POINTS

Tsukasa Mannen and Akihide Utani

Faculty of Knowledge Engineering
Tokyo City University
1-28-1, Tamazutumi, Setagaya, Tokyo 158-8557, Japan
g1781815@tcu.ac.jp

Received January 2018; accepted April 2018

Abstract. *In evolutionary computation, Differential Evolution (DE) is a method that optimizes a problem by iteratively trying to improve a candidate solution. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. As systems are enlarged, they are more difficult to handle them due to a mutual dependence of variables in optimization problems. In this paper, we put forward new methods of escaping local solution through improved DE qualities of exploration. These improvements are performed through the growing differential evolution algorithm, which changes N dynamically, and the use of ranking information to change F and CR dynamically. We conducted the experiment to compare the ordinary DE, growing DE, Ranking Differential Evolution (RDE), and proposal by using the Schwefel function and the modified Rastrigin function with a strong mutual dependence of variables.*
Keywords: Differential evolution, Growing differential evolution, Ranking differential evolution, Evolutionary computation

1. **Introduction.** The Differential Evolution (DE) algorithm [1,2] can be an effective technique classified as evolutionary computation [3] for optimization if appropriate parameters are set. DE has a potential as a solution for engineering design complications because it chooses variables at random during the crossover. DE has drawn the attention of many researchers all over the world [4-6]. For example, in [5] this algorithm is used for optimization problems of a real-time big data. In this way, DE has various possibilities. However, as systems are enlarged, they are more difficult to handle them due to a mutual dependence of variables in optimization problems.

Although this algorithm is accessible because of the few number of parameters (F: scaling factor, CR: crossover rate, N: population), it has been regarded as an issue since each problem needs different appropriate parameters and performance depends on the configuration. Additionally, falling into local solution during the search process can cause difficulties as well.

In this paper, we put forward new methods of escaping local solution through improved DE qualities of exploration. These improvements are performed through the growing differential evolution algorithm [8], which changes N dynamically, and the use of ranking information to change F and CR dynamically. The rest of the paper is organized as follows. In Section 2, we describe the outline of original DE. In Section 3, methodologies of our proposal are given. In Section 4, the experimental results are reported in detail. Finally, this paper closes with conclusions and ideas for further study in Section 5.

DOI: 10.24507/icicel.12.08.785

2. **Original DE.** Variations of DE are explained in the form of DE/base/num/cross. Base indicates how to select the base, num determines the number of vectors used for mutant vector generation and cross shows how to cross over.

In this paper, we use DE/rand/1/exp: base is chosen randomly, the number of the vectors to generate a mutant vector is 1, and exponential crossover using the probability which is decreased exponentially is applied to the crossover. The algorithm of the original DE is as follows.

• Initialization

This starts with a randomly generated population of N. The vectors are changed over generations, denoted by $t = 0, 1, 2, \ldots, t_{\max}$. Thus, the i vector solution in the population can be represented as

$$\boldsymbol{X}_{ij} = (\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2}, \boldsymbol{x}_{i3}, \ldots, \boldsymbol{x}_{iD})$$

• Mutation

Biologically, mutation is a change in the gene characteristics of a chromosome. Applied to evolutionary computation it means a change in the parameters of the vector through a perturbation with a random element. In DE literature, a parent vector from the current generation is called target vector, and the offspring obtained through recombination of target and mutant vector is called trial vector. In classic DE, to create a mutant vector for each target vector from the current population, three other distinct vectors, say, x^t_{r1}, x^t_{r2} and x^t_{r3} are selected randomly from the current population. The indices r_1, r_2 and r_3 are mutually exclusive integers randomly chosen from the range $[1, N]$, which are also different from the base vector index i. These indices are randomly generated once for each mutant vector. Now the difference of any two of these three vectors is multiplied by the scaling factor F and the scaled difference is added to the third vector to obtain the mutant vector v^t_{ij}. We can express the relation as

$$v^t_{ij} = x^t_{r_1 j} + F\left(x^t_{r2j} - x^t_{r3j}\right) \tag{1}$$

• Crossover

In exponential crossover we randomly choose an integer n between $[1, D]$. This integer n determines the index where the crossover operation would start. Another integer K is chosen from the same interval which denotes the number of parameters that would contribute to trial vector. The trial vector is generated as u^t_{ij}.

• Selection

This is the operation that determines whether the offspring represented by the trial vector or the parent represented by the target vector would survive to the next generation. The procedure is simple. If the fitness of the trial vector is better than target vector, it moves to the next generation, i.e., $t = t + 1$ else target vector is promoted.

$$x^t_i = \begin{cases} u^t_i, & f\left(u^t_i\right) < f\left(x^t_i\right) \\ x^t_i, & f\left(u^t_i\right) > f\left(x^t_i\right) \end{cases} \tag{2}$$

3. **Methodologies.** Growing DE [8] can dynamically add the number of vectors according to the stagnation of the search situation. However, in order to change the scaling factor and the crossover rate dynamically, we propose Growing Ranking Differential Evolution (Growing RDE) using an algorithm proposed in [9], which allows vectors to have ranking information. The following shows how to change parameters.

• In the case of base having good value

Mutant vectors are generated close to good vectors by decreasing F.

Increase CR so that the parents' updating speed can be increased and the offspring can get closer to the mutant vector.

- In the case of base having bad value

Generate mutant vectors in a wide range by increasing F.

Decrease CR to slow down the parents' updating speed and avoid rapid convergence.

As for the range of F and CR, we used the following values indicated in [7,9] in order not to change these two parameters.

$$F \in [0.5, 1], \quad CR \in [0.1, 1]$$

Growing DE uses a counter. If this counter exceeds a certain threshold, the algorithm judges that this exploration has fallen into the local solution and then proceeds to add a new vector into the population. This permits an escape for the population by sharing information of the new vector. The difference between ordinary DE and Growing RDE is as follows

- *Setting of counter mentioned above as C and threshold T_{int}*
- *Updating counter*

  In each generation, if the best value of this population is not changed, counter C is counted up as $C = C + 1$. If C is equal to T_{int}, the new vector is generated in this population.
- *Ranking after calculation of all vectors' value*
- *Setting of F at the stage of mutation*

  F is determined by the following equation.

$$F_i = F_{\min} + (F_{\max} - F_{\min}) \frac{(R_{r_{ij}} - 1)}{(N - 1)} \tag{3}$$

- *Setting of CR at the stage of crossover*

  CR is determined by the following equation.

$$CR_i = CR_{\max} - (CR_{\max} - CR_{\min}) \frac{(R_{r_{ij}} - 1)}{(N - 1)} \tag{4}$$

4. **Results.** We conducted the experiment to compare the ordinary DE, growing DE, RDE, and proposal by using the Schwefel function and the modified Rastrigin function with a strong mutual dependence of variables. We set common parameters as $F = 0.5$, $CR = 0.9$ and $N = 20$. In this experiment, 20000 iterations are run 50 times. As for growing DE, T_{int} is set at 50.

Figure 2 and Figure 3 show the proposal finds the optimum solution earliest. Considering concrete optimization problems, the convergence speed is an important factor.

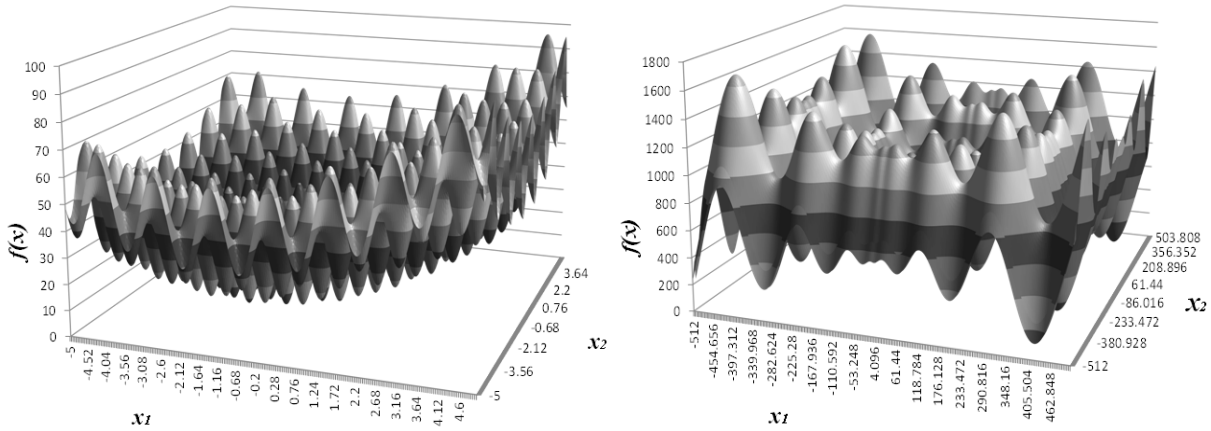

FIGURE 1. The modified Rastrigin function and the Schwefel function (Dim. = 2)

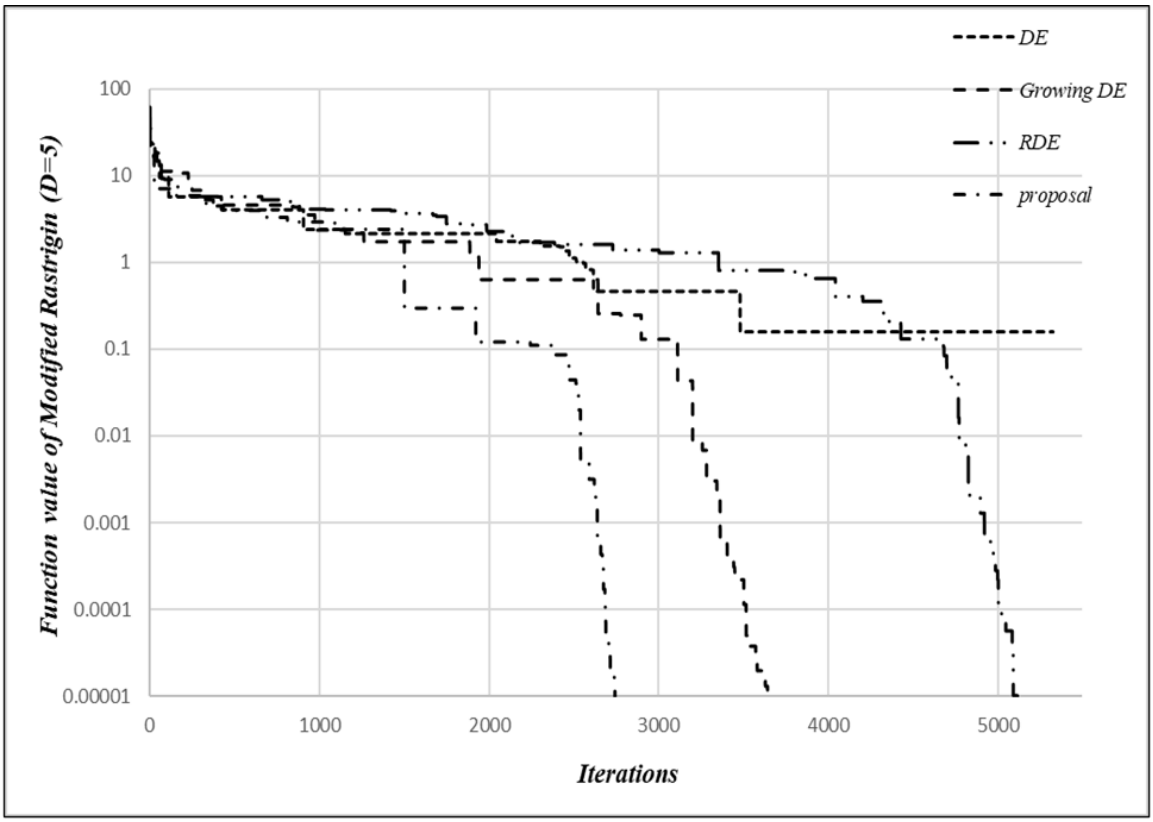

FIGURE 2. The convergence of each exploration in the modified Rastrigin function (Dim. = 5)

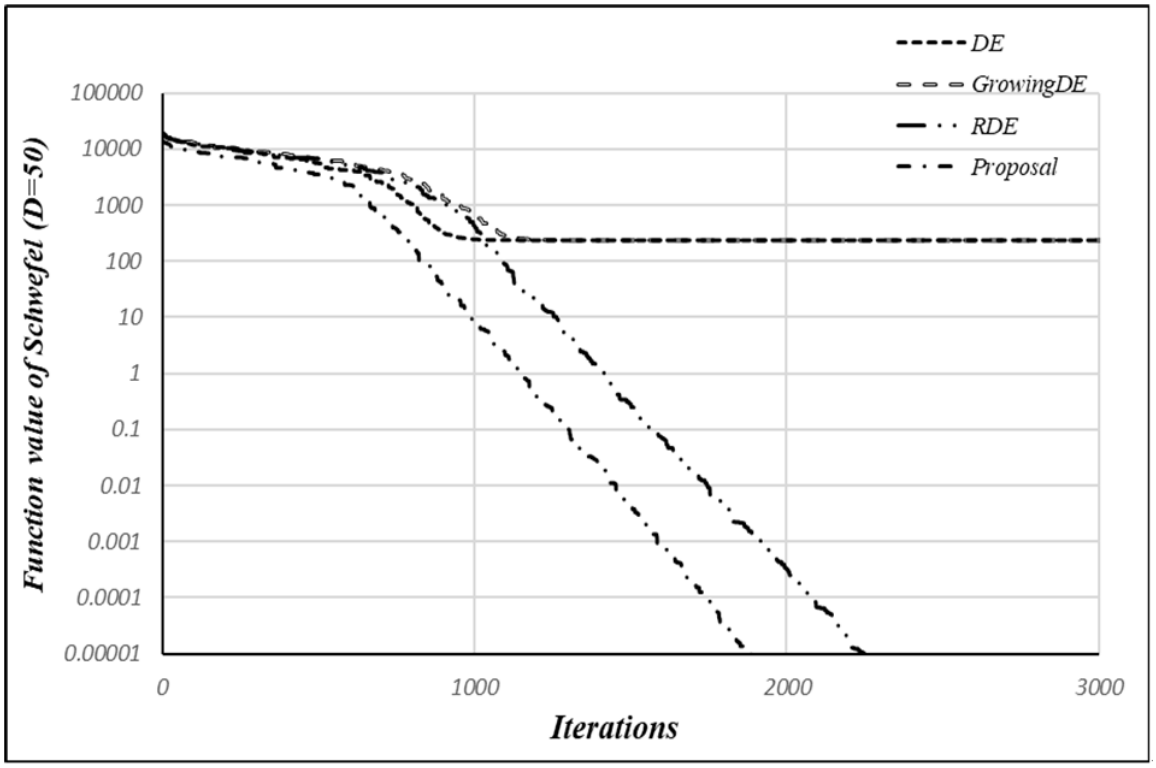

FIGURE 3. Convergence of each exploration in the Schwefel function (Dim. = 50)

TABLE 1. The number of convergence to the optimum solution in the modified Rastrigin function during the 50 trials

Algorithm	Dim. = 2	Dim. = 5	Dim. = 8
DE	48	14	0
Growing DE	47	26	2
RDE	50	22	1
Proposal	50	29	4

TABLE 2. The number of convergence to the optimum solution in the Schwefel function during the 50 trials

Algorithm	Dim. = 50	Dim. = 100	Dim. = 150
DE	6	0	0
Growing DE	15	2	0
RDE	28	24	10
Proposal	33	31	21

TABLE 3. The comparison result of each DE in the modified Rastrigin function

Algorithm	Dim.	Best	Ave.	Worst
DE	2	0.00	3.98×10^{-2}	9.95×10^{-1}
	5	0.00	7.99×10^{-1}	2.98×10^{0}
	8	1.17×10^{-1}	2.83×10^{0}	6.96×10^{0}
Growing DE	2	0.00	1.43×10^{-2}	6.99×10^{-1}
	5	0.00	3.90×10^{-1}	1.99×10^{0}
	8	0.00	2.41×10^{0}	5.97×10^{0}
RDE	2	0.00	0.00	0.00
	5	0.00	4.47×10^{-1}	1.99×10^{0}
	8	0.00	2.55×10^{0}	6.96×10^{0}
Proposal	2	0.00	0.00	0.00
	5	0.00	3.39×10^{-1}	1.99×10^{0}
	8	0.00	1.86×10^{0}	3.98×10^{0}

The larger the numbers are in Table 1 and Table 2, the more effective the algorithms are, because each number shows how many times the algorithm finds the optimum solution.

Table 3 and Table 4 exhibit that the Ave. and the Worst of the proposal are smaller than the others. (0.00 means the success in finding the solution.) This indicates that it is possible to find a value closer to the global optimum solution through searching.

5. **Conclusions.** This paper attempted to provide a new method of DE. These results show that this algorithm produces desirable value and is a better alternative in search of global optimum solution. Our proposed method also makes DE more accessible because of the way to set parameters.

This experiment has shown that several factors, such as the method of regenerating vectors that have been generated outside the domain, also affect the results. We will continue to improve the proposed algorithm by considering these factors. Furthermore, after comparing the proposed algorithm to other heuristic approaches, we will use this for concrete optimization problems, such as locating routers in sensor network. As for the

TABLE 4. The comparison result of each DE in the Schwefel function

Algorithm	Dim.	Best	Ave.	Worst
DE	50	0.00	2.81×10^2	1.07×10^3
	100	1.18×10^2	4.24×10^2	9.48×10^2
	150	2.37×10^2	7.30×10^2	1.42×10^3
Growing DE	50	0.00	1.40×10^2	5.92×10^2
	100	0.00	3.05×10^2	7.12×10^2
	150	1.18×10^2	5.00×10^2	1.06×10^3
RDE	50	0.00	7.11×10^1	4.74×10^2
	100	0.00	9.71×10^1	3.55×10^2
	150	0.00	1.68×10^2	4.74×10^2
Proposal	50	0.00	3.78×10^1	2.37×10^2
	100	0.00	4.98×10^1	2.37×10^2
	150	0.00	9.10×10^1	4.74×10^2

sensor network, these heuristic approaches also have a possibility for optimization of its power supply.

Acknowledgment. This work is partially supported by the Grant-in-Aid for Scientific Research (Grant No. 17K00349) from the Japan Society for the Promotion of Science. The authors also gratefully acknowledge the helpful comments and suggestions of the Associate Editor and reviewers.

REFERENCES

[1] R. Storn and K. Price, Minimizing the real functions of the ICEC'96 contest by differential evolution, *Proc. of the International Conference on Evolutionary Computation*, pp.842-844, 1996.
[2] R. Storn and K. Price, Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, vol.11, pp.341-359, 1997.
[3] R. Storn and K. Price, *Differential Evolution – A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*, Technical Report TR-95-012, 1995.
[4] S. Das and P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, *IEEE Trans. Evolutionary Computation*, vol.15, no.1, pp.4-31, 2011.
[5] S. Elsayed and R. Sarker, Differential evolution framework for big data optimization, *Memetic Computing*, vol.8, no.1, pp.17-33, 2016.
[6] S. Das, S. Mullick and P. N. Suganthan, Recent advances in differential evolution – An updated survey, *Swarm and Evolutionary Computation*, vol.27, 2016.
[7] J. Ronkkonen, S. Kukkonen and K. V. Price, Real-parameter optimization with differential evolution, *Proc. of 2005 IEEE Congress on Evolutionary Computation*, pp.506-513, 2005.
[8] I. Handa, C. Kurosu and T. Saito, On basic characteristics of growing differential evolution, *IEICE Technical Report*, vol.110, no.82, pp.161-164, 2010.
[9] T. Takahama, S. Sakai and A. Hara, RDE: Improving differential evolution by using ranking information of search points, *Trans. of IEICE (D)*, vol.J95-D, no.5, pp.1169-1205, 2012.