International Journal of Multimedia and Ubiquitous Engineering
Vol. 9, No. 12 (2014), pp. 241-248
http://dx.doi.org/10.14257/ijmue.2014.9.12.22

Study on Improved Differential Evolution Algorithm for Solving Complex Optimization Problem

Dao Jiang

School of Electronic and Information Engineering, Shunde Polytechnic, Shunde 528000 China
E-mail : jiangdao1979@yeah.net

Abstract

In order to improve the global searching ability of differential evolution algorithm in solving complex optimization problem, an improved differential evolution (SMDE) algorithm based on the self-adaptive method and multi-population is proposed in this paper. In the proposed SMDE algorithm, the population is divided into multi-populations in order to keep the diversity, then the self-adaptive method is used to control the parameters of differential evolution algorithm in order to balance the local search and global search ability. Finally, several complex benchmark functions are selected to validate the efficiency of the SMDE algorithm. The experiment results show that the proposed SMDE algorithm is better at the global convergence ability and the searching precision.

Keywords: *Differential evolution algorithm, Multi-populations, Self-adaptive Control method, Complex optimization problem*

1. Introduction

Evolution algorithm based on the natural biological evolution principles is a class of stochastic search and optimization methods [1]. Evolution algorithm subsequently modified a set of candidate solutions by using the two basic principles. It is proven that evolution algorithm is a class of robust and powerful search methods. However, optimization problems are more and more complex, they require high computational resources [2-4]. Differential evolution (DE) algorithm is one of the latest proposed evolution algorithms [5]. The DE algorithm is a population-based and stochastic global optimizer. In the DE algorithm, the chromosome based on floating-point vector describes the candidate solutions of the solving problem. The DE algorithm can randomly, parallel and efficiently implement the global optimization. And it takes on the strong robustness, stability and global optimization ability in solving the complex optimization problem. The DE algorithm achieves the optimization search by performing the mutation operator, crossover operator and selection operator among the current individuals. So this algorithm is similar to other evolution algorithms, such as particle swarm optimization algorithm, genetic algorithm, ant colony optimization algorithm and artificial bee colony algorithm and so on. It exists some deficiencies, for example premature convergence, the low search efficiency and local optimum. Some searchers proposed some strategies and methods for improving DE algorithm in order to solve the existed deficiencies of the DE algorithm in recent years [6-15]. Babu and Angira introduced a modification to original DE that enhances the convergence rate without compromising on solution quality. The modified differential evolution (MDE) algorithm utilizes only one set of population as against two sets in original DE algorithm at any given point of time in a generation. The results are compared with other improved DE algorithm by applying

ISSN: 1975-0080 IJMUE

benchmark test functions and five non-linear chemical engineering problems [6]. Tasgetiren, *et al.*, proposed a discrete differential evolution (DDE) algorithm to solve the no-wait flowshop scheduling problem with the total flowtime criterion. The DDE algorithm is applied to the 110 benchmark instances by treating them as the no-wait flowshop problem instances with the total flow time criterion [7], and so on.

These improved differential evolution algorithms overcome the premature convergence and falling into local optimum problems. But the DE algorithm has some pitfalls: it is slow at the exploitation of the solution, the parameters of DE are problem dependent and the choice of them is often critical for the performance of DE, choosing the best among different mutation strategies available for DE is also not easy for a specific problem. And the convergence speed and the local search ability of the differential evolution algorithm still require to be further strengthened. In order to make full use of the advantages of the self-adaptive method and multi-population, an improved differential evolution (SMDE) algorithm is proposed in this paper. To validate the effectiveness of the proposed SMDE algorithm, some Benchmark functions are selected and tested in this paper.

2. Differential Evolution

The DE algorithm is a simple, fast and efficient population-based direct search algorithm for solving global optimization problems. The DE algorithm has been widely used in many fields. The DE algorithm uses the basic framework of the genetic algorithm for designing a unique differential mutation operator. The basic operations of the DE algorithm have the mutation operation, crossover operation and selection operation. The DE algorithm uses adding weighted and random vector to mutate vectors in each step of the algorithm. It takes on some advantages of the simple structure, ease of use, robustness, and fast convergence. The DE algorithm evolves until the result or ending condition has been met. The flow of DE algorithm is shown in Figure 1.

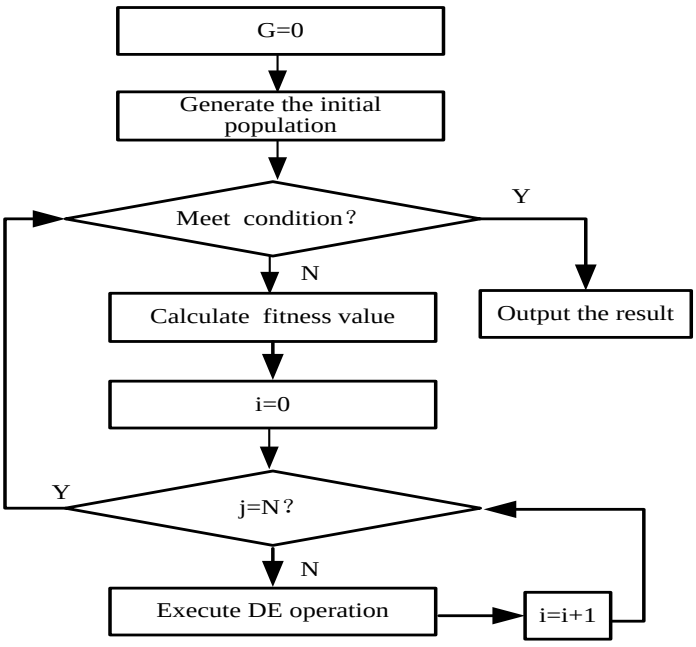

Figure 1. The Flow of the DE Algorithm

The different variants of DE algorithm will be classified by the notation $DE/\alpha/\beta/\delta$. The α is a method for selecting parent chromosome. The β is a number of difference vectors. The δ is the recombination mechanism. The different mutation and selection strategies are used to improve the convergence rate and accuracy of the DE algorithm. In this paper, the $DE/rang/1/bin$ is selected to implement the variants.

1. Generate the Initial Population

The key parameters of the population size, the mutation factor , the crossover rate and the number of iterations are initialized. The first, the population size of individuals is generated. Then the individuals are encoded. The given upper bound($x_{j\max}$) and lower bound($x_{j\min}$) of each decision variable are initialized by using the uniform probability distribution. That's $x_i = (x_1, x_2, \cdots, x_M)$ for solving N-dimension problem. The initial population is generated by:

$$x_{ji}(0) = x_{j\min} + Rand(0,1) \times (x_{j\max} - x_{j\min}) \ (\ j = 0,1,2,\cdots,D\ ) \qquad (1)$$

2. Mutation Operation

The DE algorithm uses differential strategy to realize the individual mutation of each target vector. The each target vector has two different individuals (x_{G2}, x_{G3}). There are several proposed DE algorithms.

(1)" DE/rand/1/bin"

$$x_k = x_{G1} + F \bullet (x_{G2} - x_{G3}) \qquad (2)$$

(2) "DE/rand/2/bin"

$$x_k = x_{G1} + F \bullet [(x_{G2} - x_{G3}) + (x_{G4} - x_{G5})] \qquad (3)$$

(3) "DE/best/1/bin "

$$x_k = x_{Gbest} + F \bullet (x_{G2} - x_{G3}) \qquad (4)$$

(4) "DE/rand/2/bin"

$$x_k = x_{Gbest} + F \bullet [(x_{G2} - x_{G3}) + (x_{G4} - x_{G5})] \qquad (5)$$

(5) "DE/current-to-best/1"

$$x_k = x_{G1} + F \bullet [(x_{G2} - x_{G3}) + (x_{Gbest} - x_{G1})] \qquad (6)$$

3. Crossover Operation

The crossover operation is to construct an offspring. Its essence is to do uniform crossover among the obtained individuals in the mutation operation. In this paper, the binomial cross method is selected to perform the crossover operation. To this end, a trial vector is created from the components of each mutant vector and its corresponding target vector, based on a series of binomial or exponential experiments. The specific crossover operator equation is:

$$x_{Gj} = \begin{cases} x_{kj} & rand(0,1) \le CR \\ x_{ij}^t & rand(0,1) > CR \\ j = 1,2,3,\cdots,D \end{cases} \qquad (7)$$

where $rand()$ is a uniform random number between 0 and 1. j is the variable of the j^{th}. D is variable dimension. According to the equation (7), the value of the CR is more, x_k is

more the contribution to the x_G. When CR =1, then $x_G = x_k$, it is beneficial to the rate of local search and accelerate convergence. If the value of the CR is less, x_i is more the contribution to the x_G. When CR =0, then $x_G = x_i$, it is beneficial to keeping the diversity of the population and global search.

4. Selection Operation

The DE algorithm uses the greedy search strategy to compete the obtained x_G and x_i^t after the mutation operation and crossover operation. When the fitness value of x_G is better than the fitness value of x_i^t, the x_G is selected as the offspring, otherwise, x_i^t is selected as the offspring. The selection operation equation is:

$$x_i^{t+1} = \begin{cases} x_G & f(x_G) < f(x_i^t) \\ x_i^t & f(x_G) \geq f(x_i^t) \end{cases} \quad (8)$$

3. An Improved Differential Evolution (SMDE) Algorithm

The DE algorithm is considered an efficient and powerful evolutionary algorithm. It takes on the simplicity, speed and robustness. And the performance of the DE algorithm mainly depends on the mutation operation, crossover operation, and selection operation. With the development of the society and technology, the optimization problems are becoming more and more complex. The DE algorithm and improved DE algorithm have some pitfalls, such as slow at the exploitation of the solution, difficult selecting parameters, not easy choosing the best among different mutation strategies and so on. However, for complex optimization problem, there need the different mutation strategy, crossover strategy and the values of the associated parameters in different evolution phase. So in order to make full use of the advantages of the self-adaptive method and multi-population, an improved DE (SMDE) algorithm based on the self-adaptive method and the multi-population is proposed in this paper. In the proposed SMDE algorithm, the self-adaptive method is used to automatically control the scaling and crossover factors during the running time in order to obtain the best scaling and crossover factors, balance the local search and global search, guarantee better robustness and accuracy and decrease the search time. The population is divided into multiple populations according to the fitness values of the individuals in order to keep the diversity of the premature convergence and information exchange among the sub-populations.

4. Experimental Results and Analysis

In order to test the performance of the SMDE algorithm for solving complex functions, the famous Benchmarks testing functions are used to test the performance. Some typical testing functions are selected from Benchmarks, such as Rosenbrock function, Sphere function, Ackley function, Step function, Schwefel2.21 function. The global optimum values of the selected testing functions are zero. We compare SMDE algorithm with DE algorithm, and ACDE algorithm. The parameters of these algorithms are selected after testing. The experiment environments are followed: Matlab2010b, the Pentium CPU 2.40GHz, 2.0GB RAM. The experimental parameters are set as follows: population size NP = 500, the function dimension is 30, the maximum evolution

generation $T_{\max}$ =2500, each algorithm is run independently 15 times. The specific expression and variables' range of all functions are shown in Table 1.

Table 1. Benchmarks Testing Functions

Function name	Function expression
Rosenbrock	$F_1(x)=\sum_{i=1}^{n}100(x_i-x_{i-1}^2)^2+(x_{i-1}-1)^2, \mid x_i \mid \le 30, n=30$
Sphere	$F_2(x)=\sum_{i=1}^{n}x_i^2, \mid x_i \mid \le 100, n=30$
Ackley	$F_3(x)=-20\exp[\sum_{i=1}^{n}\frac{x_i^2}{n}]-\exp(\sum_{i=1}^{n}\cos(2\pi x_i)/n)+20+e, \mid x_i \mid \le 32, n=30$
Step	$F_4(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5 \rfloor)^2, \mid x_i \mid \le 100, n=30$
Schwefel2.21	$F_5(x)=\max_i\{\mid x_i \mid, 1\le i\le n\}, \mid x_i \mid \le 30, n=30$

The experiment results with the 15 independent running are shown in Table 2.

Table 2. The Experiment Results for Benchmarks Functions

Function	Algorithm	Max value	Min value	Mean value	Standard deviation
Rosenbrock	**DE**	3.076 32e+00	7.047 20e-02	6.941 25e-01	1.811 43e+00
	ACDE	4.134 52e-05	1.268 14e-06	4.031 52e-06	3.217 92e-05
	SMDE	3.514 27e-13	3.602 18e-14	2.146 38e-14	6.392 12e-14
Sphere	**DE**	3.732 91e-13	5.204 95e-14	3.927 94e-13	5.141 68e-12
	ACDE	2.845 23e-16	9.653 47e-18	3.415 28e-17	4.615 34e-17
	SMDE	1.012 47e-28	5.227 91e-29	8.369 27e-29	6.347 13e-12
Ackley	**DE**	8.409 31e+00	2.660 96e-02	3.922 03e-02	8.385 08e-03
	ACDE	5.941 03e-09	3.524 24e-11	6.159 32e-10	3.903 21e-10
	SMDE	1.341 28e-13	1.302 26e-13	1.312 82e-13	3.468 34e-05
Step	**DE**	4.135 17e-21	7.558 43e-24	6.142 72e-22	1.231 74e-21
	ACDE	6.123 17e-25	2.105 12e-28	7.523 43e-27	5.124 02e-26
	SMDE	0.012 31e+00	0.000 00e+00	0.010 23e+00	0.002 23e+00
Schwefel2.21	**DE**	4.812 24e-02	8.624 19e-03	1.547 24e-03	1.947 29e-03

ACDE	9.312 07e-04	1.736 35e-06	1.343 41e-05	4.145 29e-04
SMDE	4.602 31e-02	3.150 21e-02	2.915 26e-02	1.490 12e-02

The performance of the SMDE algorithm is compared with other published versions of DE algorithm and ACDE algorithm. All algorithms were run 15 times for each testing function and the results presented include the maximum optimal value, the minimum optimal value, the average optimal value and the standard deviation. As can be seen in Table 2, the proposed SMDE algorithm obtained the better optimization performance than the DE algorithm for solving Rosenbrock function, Sphere function, Ackley function, Step function and Schwefel2.21 function. The proposed SMDE algorithm is better optimization performance than ACDE algorithm for solving Rosenbrock function, Sphere function, Ackley function and Step function. For Schwefel2.21 function, the ACDE algorithm can obtain better optimization solution.

In order to validate the adaptive control parameters of the SMDE algorithm, the Sphere function is used to validate and analyze the performance of the proposed algorithm in here. The experiment results are shown in Figure 2.

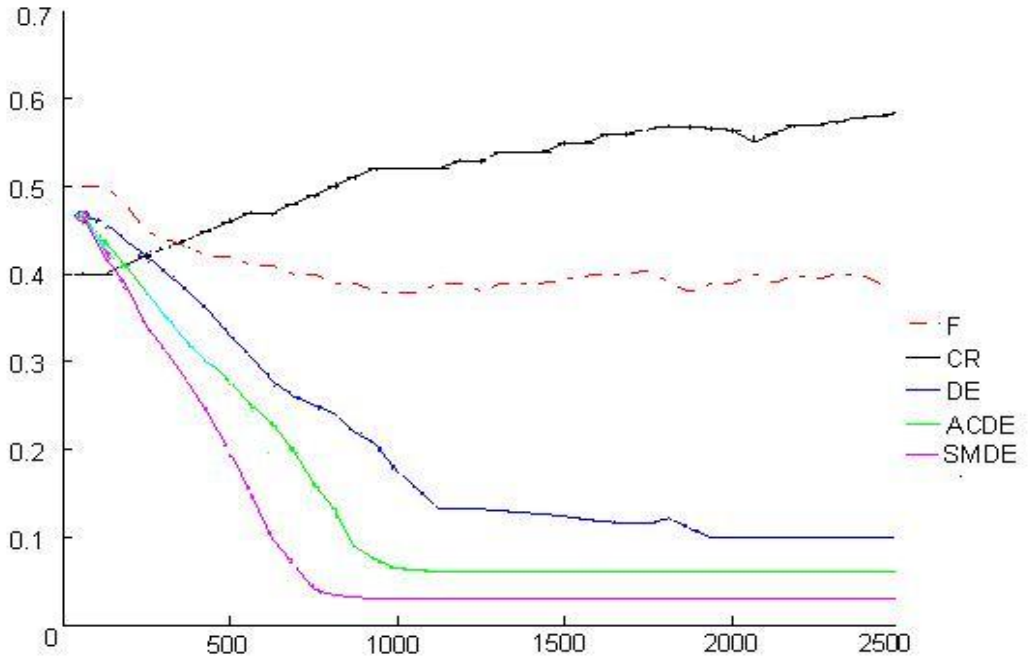

Figure 2. The Adaptive Control Parameters of the SMDE Algorithm

In proposed SMDE algorithm, the values of parameters (F and CR) is becoming in the evolution. When the number of the iteration increases, the values of parameters (F and CR) are becoming the stabilization. The variations average percentage of the DE algorithm, ACDE algorithm and SMDE algorithm are shown in Figure 2 for the trial vector strategy. We can observe that the proposed SMDE algorithm can obtain the more suitable trial vector strategy and values of parameters (F and CR) for matching different phases during the evolution. And the proposed SMDE algorithm obtained the optimization value in the least number of the iteration.

5. Conclusion

Differential evolution algorithm is a stochastic, population-based, evolutionary search algorithm. It is an efficient and powerful optimization algorithm, which widely applied in scientific research and engineering field. The performance of DE algorithm depends on the selected mutation and crossover strategies and the values of the associated control parameters. However, it is difficult to select a suitable strategy and the associated parameters, since the different optimization problems and the same optimization problem during different evolution phases should demand the different mutation and crossover strategies and the values of the associated parameters. So, for a complex optimization problem, the different mutation and crossover strategies with different parameter settings may be more effective in the process of the different evolution stages than a single mutation and crossover strategy with unique parameter settings. Therefore, the SMDE algorithm is developed in this paper. In the proposed SMDE algorithm, the population is divided into multi-populations individuals according to the difference of individuals' fitness. The self-adaptive method is used to automatically adjust the scaling factor and crossover factor during the running time. The performance of SMDE algorithm is evaluated on set of benchmark problems and is favorably compared with DE, ACDE algorithm in the literature. The results demonstrate that the proposed SMDE algorithm is overall more effective in obtaining better searching precision, convergence speed, stability, global convergence ability.

References

[1]. R. T. Arciszewski and K. De Jong, "Evolutionary computation and structural design: A survey of the state-of-the-art", Computers & structures, vol. 83, no. 23-24, **(2005)**, pp. 1943-1978.
[2]. X. Y. Yang, G. Liu, Y. X. Li and Y. Chen, "Enhancing differential evolution utilizing composite distance-based mutation operators and self-adaptive control parameters", International Journal of Advancements in Computing Technology, vol. 4, no. 12, **(2012)**, pp. 17-27.
[3]. S. W. Wang, L. X. Ding, W. N. Shu, C. W. Xie and H. Yang, "Differential evolution without scaling factor", International Journal of Advancements in Computing Technology, vol. 4, no. 4, **(2012)**, pp. 41-48.
[4]. X. F. Miao, P. G. Fan and D. J. Mu, "A new differential evolution algorithm for constrained optimization", International Journal of Advancements in Computing Technology, vol. 3, no. 10, **(2011)**, pp. 378-385.
[5]. K. P. Storn, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Optimization, vol. 11, no. 4, **(1997)**, pp. 341-359.
[6]. B. V. Babu and R. Angira, "Modified differential evolution (MDE) for optimization of non-linear chemical processes", Computers & Chemical Engineering, vol. 30, no. 6-7, **(2006)**, pp. 989-1002.
[7]. M. F. Tasgetiren, Q. K. Pan, P. N. Suganthan and Y. C. Liang, "A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flow time criterion", Proceedings of the 2007 IEEE symposium on computational intelligence in scheduling, USA, Hawaii: IEEE, **(2007)**, pp. 1-8.
[8]. R. Gamperle, S. D. Müller and P. Koumoutsakos, "A parameter study for differential evolution. Proceedings of the 2002 Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, WSEAS Press, Interlaken, Switzerland, Interlaken: WSEAS, **(2002)**, pp. 293-298.
[9]. D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms", Proceedings of the 9th International Conference on Soft Computing, Brno, **(2003)**, pp. 41-46.
[10]. S. Das, A. Konar and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search", Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, New York: ACM, **(2005)**, pp. 991-998.
[11]. U. K. Chakraborthy, S. Das and A. Konar, "Differential evolution with local neighborhood", Proceedings of the 2006 Evolutionary Computation, Vancouver, BC, Canada, **(2006)**, pp. 2042-2049.
[12]. J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm", Soft Computing, vol. 13, no. 6, **(2009)**, pp. 448-462.
[13]. J. Teo, "Exploring dynamic self-adaptive populations in differential evolution", Soft Computing, vol. 10, no. 8, **(2006)**, pp. 673-686.
[14]. L. H. Wu and Y. N. Wang, "Self-adapting control parameters modified differential evolution for trajectory planning manipulator", Control Theory Application, vol. 5, no. 4, **(2007)**, pp. 365-373.

[15]. S. C. Leandro dos, C. B. Teodoro and L. Luiz, "A Chaotic Approach of Differential Evolution Optimization Applied to Loudspeaker Design Problem", IEEE Transactions on Magnetics, vol. 48, no. 2, **(2012)**, pp. 751-754.

Author

Dao Jiang

Lecturer, received the Engineering Master degree in computer software from Jinan University in 2005, Guangzhou, China. The main research directions: Artificial Intelligence, Hybrid optimization algorithm.

