Differential Evolution Improved with Adaptive Control Parameters and Double Mutation Strategies

Jun Liu, Xiaoming Yin, and Xingsheng Gu(✉)

Key Laboratory of Advanced Control and Optimization for Chemical Processes, Ministry of Education, East China University of Science and Technology, Shanghai 200237, China
xsgu@ecust.edu.cn

Abstract. Recently, differential evolution (DE) algorithm has attracted more and more attention as an excellent and effective approach for solving numerical optimization problems. However, it is difficult to set suitable mutation strategies and control parameters. In order to solve this problem, in this paper a dynamic adaptive double-model differential evolution (DADDE) algorithm for global numerical optimization is proposed, and dynamic random search (DRS) strategy is introduced to enhance global search capability of the algorithm. The simulation results of ten benchmark show that the proposed DADDE algorithm is better than several other intelligent optimization algorithms.

Keywords: Differential evolution · Mutation strategies · Adaptive parameters · Dynamic random search

1 Introduction

Differential Evolution (DE) algorithm is a simple and efficient global optimization searching tool, firstly proposed by Storn and Price [1–4] in 1995. It has been widely used in pattern recognition [5], chemical engineering [6], image processing [7], and achieved good results. The reasons why DE has been considered as an attractive optimization method are as follows: (1) Compared with other evolutionary algorithms, DE algorithm is much simple and straightforward. (2) There are less parameters in DE. (3) Its searching is random, parallel and global. Compared with other algorithms, DE is outstanding, but the basic DE algorithm also has the disadvantages just like other intelligent algorithms. The DE algorithm suffers from the contradiction between convergence speed and accuracy, and the problem of premature convergence; additionally, it also suffers from the stagnation problem: the search process may occasionally stop proceeding toward the global optimum even though the population has not converged to a local optimum or any other point; Finally, DE algorithm is sensitive to the choice of the parameters and the same parameter is difficult to adjust to different problems [8].

In recent years, many researchers have carried out the improvement of the basic DE algorithm, which has drawn much attention. Brest et al. [9] presented the jDE algorithm in which the control parameters F and *CR* were encoded into population individuals

L. Zhang et al. (Eds.): AsiaSim 2016/SCS AutumnSim 2016, Part I, CCIS 643, pp. 186–198, 2016.
DOI: 10.1007/978-981-10-2663-8_20

and evolved with the increasing of iterations. Two new arguments were used to adjust control parameters, these arguments are calculated independently. Qin et al. [10] proposed the SaDE algorithm, in which the trial vector generation strategy was chosen from the candidate pool in accordance with the probability obtained from its success rate in generating promising results within the learning period which is a certain number of previous generations. Zhang et al. [11] presented the JADE algorithm, a new mutation strategy and an external archive were used to provide information of progress direction. This strategy is utilized to balance the greediness of the mutation and the diversity of the population. Hamzacebi et al. [12] proposed the dynamic random search (DRS) technique based on basic random search technique, DRS contain two phases: general search and local search. This technique could be applied easily in the process of optimization problem to accelerate convergence rate.

As we know, the effectiveness of basic DE in solving optimization problems mainly depends upon the selected generation (mutation and crossover operations) strategy, and associated control parameters (population size *NP*, mutation parameter *F* and crossover rate *CR*). So when solving optimization problems, the donor vector generation strategy should be determined and suitable values of control parameters needs to be chosen in advance. However, according to the characteristics of the problem and available computation resources, diverse optimization problems require different generation strategies with different control parameter values. This paper proposes a dynamic adaptive double-modle differential evolution (DADDE). In the DADDDE algorithm, the mutation mode combine two basic mutation strategies. To improve the diversity of population and balance the search capability between global and local search, the adaptive mutation parameter and crossover rate are used. In order to promote convergence rate, every iteration process is targeted at current best individual to execute dynamic random search.

The whole paper is generally organized into five parts. In the first part a brief introduction about this study is made. Following that, the second part demonstrates the process of basic differential evolution algorithm. The third part presents a dynamic adaptive double-modle differential evolution (DADDE) algorithm. In the fourth part the experimental study is taken to test the performance of DADDE compared with jDE, SaDE, JADE, PSO as well as the influence of three improvements (double-modle/adaptive parameters/dynamic random search) on DADDE. At last, the fifth part draws the conclusion of this paper.

2 Basic Differential Evolution Algorithm

The DE algorithm involves four basic operations which are called initialization, mutation, crossover and selection respectively. The whole flow chart of DE is shown in Fig. 1.

Step 1. Initialization

$G = 0, 1, 2, \ldots, G_{\max}$ denotes Generation, the ith individual $X_{i,G}$ at Gth generation is represented by:

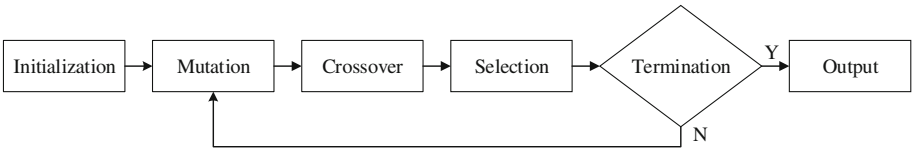

Fig. 1. Flow chart of basic DE

$$X_{i,G} = (x_{i,G}^1, x_{i,G}^2, \ldots, x_{i,G}^D), i = 1, 2, \ldots, NP \tag{1}$$

NP represents the number of population members in DE, D denotes dimension. The search space of uniformly and randomly distributed individuals constrained by the prescribed minimum and maximum bounds $[X_{\min}, X_{\max}]$, $X_{\min} = (x_{\min}^1, x_{\min}^2, \ldots, x_{\min}^D)$, $X_{\max} = (x_{\max}^1, x_{\max}^2, \ldots, x_{\max}^D)$. When Generation $G = 0$, the initial population is formed by individuals generate in $[X_{\min}, X_{\max}]$.

$$X_{i,0} = (x_{i,0}^1, x_{i,0}^2, \ldots, x_{i,0}^D), i = 1, 2, \ldots, NP \tag{2}$$

Therefore the jth component of the ith individual should be initialized as $x_{i,0}^j = x_{\min}^j + rand(0,1) \cdot (x_{\max}^j - x_{\min}^j)$, $j = 1, 2, \ldots, D$, $rand[0,1]$ is a uniformly distributed random number within $[0,1]$.

Step 2. Mutation

Mutation operation contains variant forms. The general process of mutation is expressed by

$$V_{i,G} = X_{r_1,G} + F(X_{r_2,G} - X_{r_3,G}) \tag{3}$$

where i means the ith individual vector of current generation. $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$ are three different random integers, besides each one of them should be different from i. $V_{i,G}$ denotes donor vector. The mutation control parameter F is a real and constant factor that controls the amplification of the differential variation.. If $V_{i,G}$ is not within $[X_{\min}, X_{\max}]$, let $V_{i,G} = X_{\min} + rand(0,1) \cdot (X_{\max} - X_{\min})$, $rand[0,1]$ is a uniformly distributed number randomly chosen from $[0,1]$.

Step 3. Crossover

The operands of crossover are components of the individual. Through this operation, the donor vector $V_{i,G}$ exchanges its components with the target vector $X_{i,G}$ to form the trial vector $U_{i,G} = (u_{i,G}^1, u_{i,G}^2, \ldots, u_{i,G}^D), i = 1, 2, \ldots, NP$

$$u_{i,G}^j = \begin{cases} v_{i,G}^j, & rand_j \le CR \, or \, j = randn_j \\ x_{i,G}^j, & rand_j > CR \; and \; j \ne randn_j \end{cases}, j = 1, 2, \ldots, D \tag{4}$$

$rand_j$ is a number randomly chosen from $[0,1]$, $randn_j$ is a randomly chosen index from $\{1, 2, \ldots, D\}$. The crossover control parameter CR is a real and constant factor that

controls which parameter contributes to which trial vector parameter in the crossover operation, ranging between [0,1].

Step 4. Selection

Selection is based on Greedy policy. The offspring vector is acquired through comparing the fitness value of the trial vector $U_{i,G}$ and target vector $X_{i,G}$ according to

$$X_{i,G+1} = \begin{cases} U_{i,G}, f(U_{i,G}) < f(X_{i,G}) \\ X_{i,G}, f(U_{i,G}) \geq f(X_{i,G}) \end{cases}, \ i = 1, 2, \ldots, NP \tag{5}$$

f is the function of the fitness value. The one which has the better value between $U_{i,G}$ and $X_{i,G}$ should be chosen as the new individual, then add one to generation G. Equation (5) is for dealing with the minimization.

Step 5. Termination

If the population meet the termination conditions or reach the upper limit of generation, Output the optimal solution. Otherwise, go to step 2 till meet the termination conditions.

3 Dynamic Adaptive Double-Model Differential Evolution

This paper proposes a dynamic adaptive double-model differential evolution (DADDE). In DADDE, the mutation mode combine two basic mutation strategies. To improve the diversity of population and balance the search capability between global and local search, the adaptive mutation parameter and crossover rate are used. In order to promote convergence rate, every iteration process is targeted at current best individual to execute dynamic random search. These improvements to basic DE are as the following.

3.1 Double Mutation Strategies

According to the DE algorithm which was firstly proposed by Storn and Price [1–4], there are ten kinds of basic mutation strategies, these strategies are provided in Table 1.

In general, DE/x/y/z denotes different mutation strategies. DE denotes differential evolution, x denotes base vector which contain rand, best, rand-to-best, current-to-best

Table 1. Mutation strategies of DE

Number	Mutation strategies	Number	Mutation strategies
1	DE/best/1/exp	6	DE/best/1/bin
2	DE/rand/1/exp	7	DE/rand/1/bin
3	DE/rand-to-best/1/exp	8	DE/rand-to-best/1/bin
4	DE/best/2/exp	9	DE/best/2/bin
5	DE/rand/2/exp	10	DE/rand/2/bin

and so on. y denotes the number of differential vectors. z denotes crossover strategies which include exponential crossover and binomial crossover.

Each strategies has its own characteristics, but through a large number of studies, Storn and Price found that $DE/rand/1/bin$ and $DE/best/2/bin$ have better performance, also have been applied to the practical industrial process mostly. $DE/rand/1/bin$ is expressed as Eq. (3), $DE/best/2/bin$ is expressed as

$$V_{i,G} = X_{best,G} + F[(X_{r_1,G} - X_{r_2,G}) + (X_{r_3,G} - X_{r_4,G})] \tag{6}$$

$X_{best,G}$ denotes the best individual in current population. In order to make full use of the better global search capability of $DE/rand/1/bin$ and the better convergence ability of $DE/best/2/bin$, Overcome the disadvantages of both strategies as well, Hu [14] combined these two mutations as follows:

$$V_{i,G} = \begin{cases} X_{r_1,G} + F(X_{r_2,G} - X_{r_3,G}), \text{ if } rand \geq \sqrt{\frac{G}{Gm}} \\ X_{best,G} + F[(X_{r_1,G} - X_{r_2,G}) + (X_{r_3,G} - X_{r_4,G})], \text{ otherwise} \end{cases} \tag{7}$$

The threshold value $\varphi = \sqrt{\frac{G}{Gm}}$, is an variable increase with the growth of generation. Here we set a new threshold value:

$$\varphi = \sqrt{\frac{G}{Gm}} \cdot (\varphi_{\max} - \varphi_{\min}) + \varphi_{\min} \tag{8}$$

$[\varphi_{\min}, \varphi_{\max}] = [0.1, 1]$. At the beginning, $DE/rand/1/bin$ will be used much more, as generation increase, algorithm will use $DE/best/2/bin$ more often.

3.2 Adaptive Mutation Parameter and Crossover Rate

Adaptive parameter will achieve a balance between the convergence speed and global search ability. when F have a large value, global optimization ability will be stronger, but convergence rate become slower. A large value of CR will lead to better convergence speed, worse stability and lower success rate of the algorithm, premature convergence become more obvious as well. In order to prevent the occurrence of premature convergence and guarantee fast convergence speed at the same time, taking the follow adaptive mechanism is to assign the parameters.

$$F = F_{\max} - (F_{\max} - F_{\min}) \cdot \sqrt{\frac{G}{Gm}} \tag{9}$$

$$CR = CR_{\min} + (CR_{\max} - CR_{\min}) \cdot \sqrt{\frac{G}{Gm}} \tag{10}$$

In DADDE, $\mathrm{F} \in [0.4, 0.9], \mathrm{CR} \in [0.6, 0.9]$. With the increase of iteration, F increase and CR decrease, to insure the diversity of population and global search

capability at the beginning of the algorithm, to reduce the diversity and promote the algorithm convergence in the later stage of the algorithm.

3.3 Dynamic Random Search

Dynamic random search (DRS) technique is based on searching the solution space randomly to acquire the best value of minimization problem. DRS contain two phases: general search and local search. DRS is simple and easily adaptable for any problems. Because of these two essential advantages, this technique could be applied easily in the process of optimization problem to accelerate convergence speed. Steps of local search phase [12] which is added into basic DE algorithm for soluting continuous minimization problem are as follows (objective function of the problem is described as f(x)) (Fig. 2).

Step 1. Set the initial values: N donotes maximum iteration, E denotes stop criterion, α_k denotes level counter, $k = 0$, epoch = 0, $X_{current} = X_{best}$, $\alpha_k = X_{best}$.

Step 2. Reset the iteration counter, $n = 0$.

Step 3. Generate dX random vector within the range $[-\alpha_k, \alpha_k]$.

Step 4. Update epoch, epoch = epoch + 1.

Step 5. $f_{new} = f(X_{current} + dX)$

if $f_{new} < f_{best}$, then $f_{best} = f_{new}$, $X_{best} = X_{current} + dX$.

Increase iteration number by one, $n = n + 1$. Go to step 7.

if $f_{new} < f_{current}$, then $f_{current} = f_{new}$, $X_{current} = X_{current} + dX$.

Increase iteration number by one, $n = n + 1$. Go to step 7.

Step 6. $f_{new} = f(X_{current} - dX)$

if $f_{new} < f_{best}$, then $f_{best} = f_{new}$, $X_{best} = X_{current} - dX$.

Increase iteration number by one, $n = n + 1$. Go to step 7.

if $f_{new} < f_{current}$, then $f_{current} = f_{new}$, $X_{current} = X_{current} - dX$.

Increase iteration number by one, $n = n + 1$. Go to step 7.

Step 7. If iteration counter is less than its maximum value ($n < N$), then go to step 3.

Step 8. $k = k + 1$.

Step 9. $\alpha_k = \alpha_{k-1} * 0.5$

Step 10. If stop criterion is reached (epoch = E), then quit. Otherwise, go to step 2.

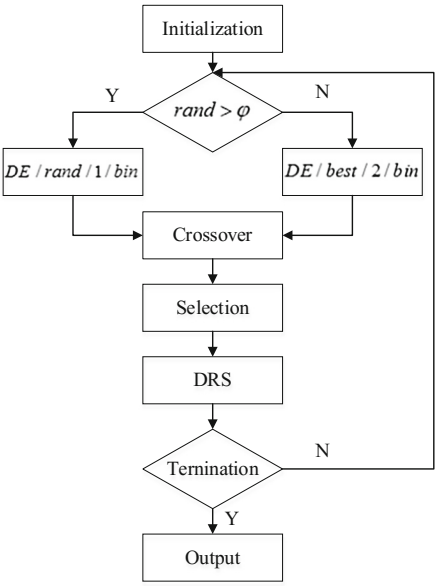

Fig. 2. Flow chart of DADDE

4 Experimental Study

4.1 Benchmark Functions

In this section, 10 global minimization benchmark functions are presented to evaluate the performance of the proposed DADDE algorithm against other intelligent algorithms. These functions (f1–f10) are dimension-wise scalable [15]. Among these benchmark functions, f1–f6 represent unimodal functions, f7–f10 represent multimodal functions. The value of dimension, names, optimum value, and initialization ranges for these benchmark functions are provided in Table 2.

4.2 Experimental Setup

The proposed DADDE algorithm was compared with various outstanding algorithms such as PSO, jDE, JADE and SaDE, to test the performance of DADDE. The experiments were conducted on the suite of 10 test functions listed in Table 4. For all the algorithms, the maximum number of function evaluations was set to 150,000 generations and the population size was set as $NP = 100$. Other parameters in PSO, jDE, JADE and SaDE, were set based on previous literature [9–11]. Every algorithm ran 30 times on the 10 test functions, the optimal values, the average values and standard deviation of the functions were obtained in 30 runs. The optimal value and the average value can show the quality of the solution obtained from the algorithm, and the standard deviation can be used to explain the stability and robustness of the algorithm.

Table 2. Benchmark functions

Function	Name	Dimension	f (x*)	Initial range
f1	Sphere	30	0	$[-100, 100]^D$
f2	Schwefel 2.22	30	0	$[-10, 10]^D$
f3	Schwefel 1.2	30	0	$[-100, 100]^D$
f4	Schwefel 2.21	30	0	$[-100, 100]^D$
f5	Rosenbrock	30	0	$[-30, 30]^D$
f6	Quartic	30	0	$[-1.28, 1.28]^D$
f7	Schwefel 2.26	30	-12569.5	$[-500, 500]^D$
f8	Rastrigin	30	0	$[-5.12, 5.12]^D$
f9	Ackley	30	0	$[-32, 32]^D$
f10	Griewank	30	0	$[-600, 600]^D$

Before the experiment, in order to illustrate the effectiveness of the repetition of previous algorithm code, a test had been taken. In this test, all the parameters including population size, the maximum number of evaluation, running times were set up as same as the original reference. The results of this test and from the original literature were almost in the same order of magnitude. For example, in literatures [52] parameters of jDE were set as: $\tau_1 = \tau_2 = 0.1$, initialization of F and CR is 0.5, the maximum number of function evaluations is 1500. jDE algorithm ran 50 times on f1, the average values was 1.10E-28 and standard deviation was 1.00E-28 according to the original literature. The results of the code edited in this study showed that the average value was 8.97E-27 and the standard deviation was 4.66E-27. This test proved that the code used in this paper can reflect the performance of previous algorithms, so as to ensure the effectiveness of comparison between DADDE and other algorithms.

4.3 Comparison Between DADDE and Other Algorithms

Table 3 presents the results over 30 independent runs on 10 test functions. Wilcoxon's rank sum test at a 0.05 significance was conducted between DADDE and each of PSO, jDE, JADE and SaDE. Moreover, "+", "-" and "≈" in Table 4 denote that the performance of DADDE is better than, worse than, and similar to that of the corresponding algorithm respectively. Results of comparison based on Wilcoxon's test can be directly observed from Table 4.

It is obviously that the proposed DADDE algorithm performed better than the other compared algorithms. For example, it was better than PSO on all 10 test functions, better than jDE on 7 test function sand similar to it on 2 test functions, better than SaDE on 7 test functions and similar to it on 3 test functions, better than JADE on 6 test functions and similar to it on 3 test functions.

Table 3. Experimental results of 10 benchmark functions

Function	Algorithm	Best	Mean	Std
f1	PSO +	1.37E-012	8.07E-033	3.35E-032
	jDE +	9.24E-067	6.34E-065	2.67E-064
	SaDE +	2.34E-046	4.93E-042	1.17E-043
	JADE ≈	9.37E-065	8.02E-060	2.26E-059
	DADDE	0.00E+000	0.00E+000	0.00E+000
f2	PSO +	1.98E-014	4.12E-013	2.87E-013
	jDE +	9.60E-044	4.57E-043	4.25E-042
	SaDE +	9.38E-021	8.03E-020	6.01E-020
	JADE +	3.95E-035	2.88E-034	3.98E-035
	DADDE	0.00E+000	0.00E+000	0.00E+000
f3	PSO +	1.78E+000	2.86E+000	6.34E+000
	jDE +	3.56E-007	5.21E-007	7.75E-007
	SaDE +	2.45E-003	4.82E-003	7.24E-003
	JADE ≈	1.66E-037	4.33E-037	1.26E-036
	DADDE	4.03E-269	3.63E-250	1.73E-249
f4	PSO +	1.39E+000	2.39E+000	4.32E+000
	jDE +	2.64E-001	1.74E-001	5.62E-001
	SaDE +	4.15E-002	4.77E-002	1.60E-002
	JADE +	5.61E-011	2.26E-011	4.70E-011
	DADDE	1.22E-274	1.56E-252	3.24E-251
f5	PSO +	1.20E+000	2.13E+000	4.17E+000
	jDE +	1.35E-004	7.57E-004	2.23E-004
	SaDE +	4.58E-004	9.66E-004	5.82E-004
	JADE +	3.69E-022	4.90E-021	9.41E-021
	DADDE	0.00E+000	0.00E+000	3.24E+000
f6	PSO +	4.39E-002	5.12E-002	1.71E-002
	jDE ≈	2.81E-003	3.43E-003	2.11E-003
	SaDE ≈	6.40E-003	7.06E-003	3.71E-003
	JADE -	1.93E-003	2.43E-003	1.18E-003
	DADDE	2.64E-003	2.94E-003	1.97E-003
f7	PSO +	1.25E+004	1.20E+004	2.81E+002
	jDE ≈	1.25E+004	1.25E + 004	6.77E+001
	SaDE ≈	1.25E+004	1.25E+004	9.47E+001
	JADE +	1.25E+004	1.22E+004	1.93E+002
	DADDE	1.25E+004	1.25E+004	0.00E+000
f8	PSO +	2.93E-012	1.86E-011	1.08E-011
	jDE +	1.68E-013	2.44E-012	3.83E-012
	SaDE +	1.88E-013	2.78E-012	4.06E-012
	JADE +	3.11E-013	8.98E-0.12	5.12E-012
	DADDE	0.00E+000	0.00E+000	0.00E+000

(*Continued*)

Table 3. (*Continued*)

Function	Algorithm	Best	Mean	Std
f9	PSO +	1.19E-008	2.98E-008	3.88E-008
	jDE -	5.16E-017	6.11E-017	1.25E-017
	SaDE ≈	1.29E-013	3.07E-013	2.41E-013
	JADE ≈	8.64E-015	8.06E-015	6.53E-015
	DADDE	3.28E-015	4.44E-015	2.20E-015
f10	PSO +	1.98E-002	4.12E-002	2.87E-002
	jDE +	1.57E-006	1.97E-006	4.06E-006
	SaDE +	1.55E-005	2.20E-005	1.76E-005
	JADE +	1.75E-006	2.27E-006	4.40E-006
	DADDE	0.00E+000	0.00E+000	0.00E+000

Table 4. Comparison results based on Wilcoxon's rank sum test.

Function	PSO	jDE	SaDE	JADE
DADDE better	10	7	7	6
DADDE worse	0	1	0	1
DADDE equal	0	2	3	3
Success Rate	100 %	90 %	100 %	90 %

4.4 Comparison of the Influences on DADDE with or Without Double-Modle/Adaptive Parameters/Dynamic Random Search

The proposed algorithm is tested to prove that the global search capabilities of DADDE can be enhanced after three improvements (double-modle/adaptive parameters/dynamic random search) are added. For convenience, the algorithm without adaptive parameters and dynamic random search is called DDE, the algorithm without dynamic random search is called ADDE.

In Fig. 3, the convergence graphs show the fitness of function from DE, DDE, ADDE and DADDE on two representative benchmark functions (f4, f9) with $D = 30$, $NP = 100$ and FES = 1500. The convergence speed of DADDE was the best one. Table 4 presents the results after these four algorithms ran 30 times. It can be known that the average values and standard deviation of DADDE were both relatively less than that of others.

According to the evidence provided by Fig. 3 and Table 5, the convergence rate and accuracy of DADDE were better than the other three comparisons, so it came to a conclusion that global search capabilities of DADDE can be enhanced by these presented improvements.

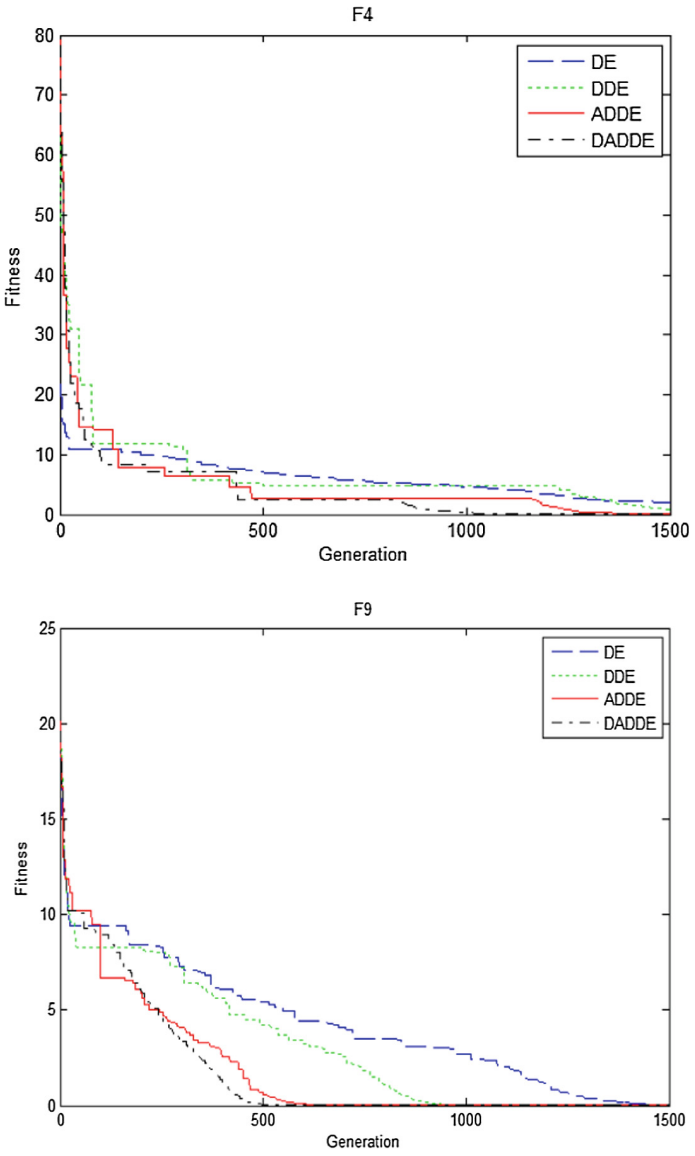

Fig. 3. The convergence graphs for best fitness

This improved algorithm can balance the search capability between global and local search. But by using double mutation strategies and adaptive parameters, good global search capabilities are achieved at the cost of reduction of convergence rate. Although dynamic random search is added to promote convergence rate, on several test functions experimental convergence speed were still influenced, this limitation is more obvious on unimodal functions.

Table 5. Experimental results of DE, DDE, ADDE and DADDE

Function	Algorithm	Best	Mean	Std
f4	DE	4.00E + 00	5.18E + 00	1.39E + 00
	DDE	5.64E-01	8.57E-01	5.17E-01
	ADDE	3.42E-02	4.89E-02	6.92E-02
	DADDE	7.35E-04	6.14E-03	2.08E-03
f9	DE	3.34E-02	4.11E-02	2.87E-02
	DDE	2.67E-10	3.90E-10	4.01E-10
	ADDE	7.93E-13	3.08E-12	5.62E-12
	DADDE	5.18E-15	7.31E-15	4.55E-15

5 Conclusions

As an excellent and efficient search and optimization algorithm, differential evolution (DE) has been widely applied in science and engineering. In the DE algorithm, mutation strategies and control parameters are very significant to the algorithm's performance. However, it is difficult to select a befitting strategy and parameters. Moreover, dynamic random search could be applied easily in the process of optimization problem to accelerate convergence rate. Therefore, a DADDE algorithm is put forward to improve the performance of basic DE.

In this paper, the experimental studies had been executed on ten global numerical optimization functions adopted from previous literature. DADDE was compared with other four advanced optimization algorithms, such as PSO, jDE, SaDE and JADE. The experimental results indicated that the performance of DADDDE was better than the other four algorithms totally. In order to prove that the global search capabilities of basic DE can be enhanced by these three improvements made in DADDE, DADDE was compared with the DE, DDE and ADDE. All of the experimental results showed that the performance of DADDE was more outstanding than other competitors.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Grant No. 61573144, 61174040), Shanghai Commission of Science and Technology (Grant no. 12JC1403400), and the Fundamental Research Funds for the Central Universities.

References

1. Storn, R., Price, K.V.: Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces, ICSI, USA, Technical Report TR-95–012 (1995)
2. Storn, R., Price, K.V.: Minimizing the real functions of the ICEC 1996 contest by differential evolution. In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation, pp. 842–844 (1996)
3. Storn, R.: On the usage of differential evolution for function optimization. In: Proceedings of the North American Fuzzy Information Processing Society Conference, pp. 519–523 (1996)

4. Storn, R., Price, K.V.: Differential evolution: A simple and efficient heuristic for global optimization over continuous space. J. Global Optim. **11**(4), 341–359 (1997)
5. Secmen, M., Tasgetiren, M.F.: Ensemble of differential evolution algorithms for electromagnetic target recognition problem. IET Radar Sonar Navig. **7**(7), 780–788 (2013)
6. Sharma, S., Rangaiah, G.P.: An improved multi-objective differential evolution with a termination criterion for optimizing chemical processes. Comput. Chem. Eng. **56**, 155–173 (2013)
7. Zhu, J.X., Wen, X.B., Xu, H.X., Sun, L.Q.: Image sparse decomposition and reconstruction based on differential evolution algorithm. Adv. Inf. Sci. Serv. Sci. **3**(10), 131–137 (2011)
8. Daniela, Z.: Influence of crossover on the behavior of differential evolution algorithms. Appl. Soft Comput. **9**(3), 1126–1138 (2009)
9. Brest, J., Mernik, M.: Population size reduction for the differential evolution algorithm. Appl. Intell. **29**(3), 228–247 (2008)
10. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2005), pp. 1785–1791. IEEE Press, Edinburgh, Scotland (2005)
11. Zhang, J.Q., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
12. Hamzacebi, C., Kutay, F.: Continuous functions minimization by dynamic random search technique. Appl. Math. Model. **31**(10), 2189–2198 (2007)
13. Hamzacebi, C., Kutay, F.: A heuristic approach for finding the global minimum: adaptive random search technique. Appl. Math. Comput. **173**(2), 1323–1333 (2006)
14. Hu, Z.Q.: The optimization of differential evolution algorithm and its application research, pp. 28–30 (2013). (in Chinese)
15. Yao, X., Liu, Y., Lin, G.M.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)