Self-adapting Differential Evolution Algorithm with Extension Variable Dimension

FENG Da[1], GAO Yuan[2], GAO LiQun[3]

1. AVIC Aerodynamics Research Institute, Shenyang, 110034, China
E-mail: fengdafd@126.com

2. Department of Computer Science, The University of Western Ontario, Ontario, N6A 5B7, Canada
E-mail: ygao72@csd.uwo.ca

3. School of Information Science and Engineering, Northeastern University, Shenyang, 110004, China
E-mail: gaoliqun@mail.neu.edu.cn

Abstract: Recently, the differential evolution (DE) algorithm has attracted much attention as an effective approach for numerical optimization. Since the performance of DE is sensitive to the choice of associated control parameters, a large number of strategies on parameter determination have been presented in the past several years. However, most of them have limitations. Thus, to get optimal performance, time-consuming parameter tuning is necessary. This paper introduces an extension variable dimension of DE (EVSDE). In EVSDE, the control parameters are considered as a variable of the component. The variable dimension is expended and a new mutation strategy is employed for the extension dimension of variables on mutation operation. On the basis of experience value, the control parameters follow the individual variable and implement the dynamic self-adaptive process in the evolutionary process. It thus helps to improve the robustness of the algorithm and avoid premature convergence. Simulation results show the EVSDE is better than or at least comparable to other classic and adaptive DE algorithms from the literature in terms of convergence performance for a set of 10 benchmark problems.

Key Words: Differential evolution algorithm, Adaptive parameter control, Extension variable dimension

1 INTRODUCTION

Unconstrained optimization problems are often described as the following form. Give a real-valued objective function f, define x as the independent variable of function f ($x \in \Omega \subset R^D$, x is a continuous variable, D is the dimension of the problem), and find the most optimization x^* such that the corresponding objective function value f^* is minimized.

$$f^* = f(x^*) = \min\{f(x) \mid x \in \Omega\} \quad (1)$$

x is a vector defined as $x=(x_1, x_2, \cdots, x_D) \in \Omega$. The domain Ω is defined by specifying the lower ($x_{j,\min}$) and upper ($x_{j,\max}$) limits of each dimension j. Therefore, x_j is bounded, i.e. $x_{j,\min} \leqslant x_j \leqslant x_{j,\max}$. Unconstrained optimization problems are frequently encountered in many areas such as scientific research and economic management. The global optimization problems, which are required to locate the global minimum among many local minima, are non-differentiable, noisy and simulation based. Hence the gradient-based methods cannot be used for finding the global minimum of such problems [1].

The evolutionary algorithms (EAs) inspired by Darwin's theory of evolution are well known for their capability to deal with non-linear and complex optimization problems by simulating natural selection and biological evolution. Generally speaking, the evolution algorithms consist of the following two steps.

1. Initialization: A population of individuals is randomly initialized, and we can regard each individual as a potential solution to the optimization problem. The fitness function is used for the quality evaluation of each solution.

2. Mutation and Crossover: The selection process is apt to the preferable individuals to increase their chances of being included in the next population. This procedure is repeated until convergence is reached. Step 2 is repeated until the convergence is reached. The best solution is expected to be a near-optimum solution [2].

As one of the evolution algorithms, differential evolution (DE) proposed by R. Storn and K. Price [3] is a fast and simple technique which performs well on a wide variety of problems. However, the performance of DE algorithm is still very sensitive to the setting of control parameters such as the mutation factor (F) and the crossover rate (CR), which may significantly influence the searching accuracy and convergence speed of the DE algorithm [4]. Recently, researchers have worked on setting the control parameters and came up with many improved parameter adaptation mechanisms. They can be classified into three classes of parameter control mechanisms: deterministic parameter control[5], adaptive parameter control[6] and self-adaptive parameter control[7]. If adaptive or self-adaptive parameter control is well designed, the robustness of the DE algorithm and convergence rate can be enhanced.

The rest of this paper is organized as follows: Section 2 gives a brief introduction of the DE algorithm; Section 3 discusses the EVSDE algorithm; one experiment is presented in the Section 4 and the simulation results show that EVSDE is better than or at least comparable to other classic and adaptive DE algorithms; some sentiments are given in Section 5.

2 BRIEF INTRODUCTION OF DE ALGORITHM

Differential evolution algorithm adopts real-number encoding, and follows the general procedure of an evolutionary algorithm (EA). It shares a common terminology of mutation, crossover and selection operators with the GA. In selection strategy, DE usually applies

*This work is supported by National Natural Science Foundation (NNSF) of China under Grant 60674021

tournament scheme, and the mode of crossover is roughly the same as the genetic algorithm, but DE employs differential strategy in mutation and disturbs individuals by differential vector of mutation operator. The DE works as the following four steps. First, all individuals are initialized with uniformly distributed random numbers and evaluated with the fitness function provided. Then the following are executed until a stopping criterion is satisfied [8].

The problem of minimizing function $f(x_1, x_1, \cdots, x_D)$ is usually described as

$$\min_{x\in\Omega} f(x_1, x_2, ..., x_D), \quad \text{s.t. } x_{j,\min} \le x_j \le x_{j,\max} \quad j = 1, 2, ..., D \tag{2}$$

Where D is the dimension of the problem, $x_{j,\min}$ and $x_{j,\max}$ represents the lower and upper bounds of the j-th component, respectively. The detailed working process of DE is described as below.

2.1 Initialization

Assume that the amount of individuals is N in the optimization problem and g is the current generation index. A population of ND-dimensional vectors $x^i_g=\{x^i_{1,g}, x^i_{2,g}, \cdots, x^i_{D,g}\}$($i$=1,2, ⋯, N)is initialized by distributing with random uniformity between the pre-specified lower and upper bounds, $x^i_{g,\min}=\{x^i_{1,g,\min}, x^i_{2,g,\min}, \cdots, x^i_{D,g,\min}\}$ and $x^i_{g,\max}=\{x^i_{1,g,\max}, x^i_{2,g,\max}, \cdots, x^i_{D,g,\max}\}$, respectively. Take the j-th component of the i-th individual for example:

$$x^i_{j,0} = rand_j \cdot \left(x^i_{j,0,\max} - x^i_{j,0,\min}\right) + x^i_{j,0,\min} \tag{3}$$

where i=1,2, ⋯,N and j=1,2, ⋯,D, g is the generation index. A uniform distribution between 0 and 1 is generated from $rand_j$.

2.2 Mutation

According to the classical mutation strategy, a mutation vector $v^i_g(v^i_g=\{v^i_{1,g}, v^i_{2,g}, \cdots, v^i_{D,g}\}, i$=1,2, ⋯,$N$) is produced after mutation through differential strategy, which selects two different individuals from population and multiplies (x^{r2}_g-x^{r3}_g) with mutation factor F to add to x^{r1}_g [9]. r_1, r_2 and $r_3\in$ {1,2, ⋯,N} are randomly chosen integers, different from each other and also different from the running index i. The progress can be expressed as

$$v^i_g = x^{r1}_g + F\cdot(x^{r2}_g - x^{r3}_g) \tag{4}$$

The most often-used mutation strategies implemented are listed below [5].

a)"DE/rand/1":

$$v^i_g = x^{r1}_g + F(x^{r2}_g - x^{r3}_g) \tag{5}$$

b)"DE/best/1":

$$v^i_g = x^{best}_g + F(x^{r1}_g - x^{r2}_g) \tag{6}$$

c)"DE/current-to-best/1":

$$v^i_g = x^i_g + F(x^{best}_g - x^i_g) + F(x^{r1}_g - x^{r2}_g) \tag{7}$$

d)"DE/best/2":

$$v^i_g = x^{best}_g + F(x^{r1}_g - x^{r2}_g) + F(x^{r3}_g - x^{r4}_g) \tag{8}$$

e)"DE/rand/2":

$$v^i_g = x^{r1}_g + F\left(x^{r2}_g - x^{r3}_g\right) + F\left(x^{r4}_g - x^{r5}_g\right) \tag{9}$$

where r_1, r_2, ⋯, $r_5\in$ {1,2, ⋯,N} are randomly chosen integers which are different from each other and also different from the running index i, and x^{best}_g is the best individual vector with the best fitness value in the population at the g-th generation.

2.3 Crossover

After the mutation phase, crossover is performed between the target vector x^i_g and the mutated vector v^i_g to generate a trial vector $u^i_g(u^i_g=\{u^i_{1,g}, u^i_{2,g}, \cdots, u^i_{D,g}\}, i$=1,2, ⋯, N) for the next generation. Crossover is introduced to increase the diversity of the population [10]. Formula 10 is the definition of binomial crossover.

$$u^i_{j,g} = \begin{cases} v^i_{j,g}, & rand_j \le CR \text{ or } j = j_{rand} \\ x^i_{j,g}, & otherwise(j = 1, 2, ..., D) \end{cases} \tag{10}$$

Where CR is the crossover rate, and j_{rand} is a randomly chosen integer in the range [1, D].

2.4 Selection

The selection operation is to pick up the better one from the trial vector u^i_g and the target vector x^i_g of current population according to the greedy selection scheme. The better one will survive at the next generation:

$$x^i_{g+1} = \begin{cases} u^i_g, & f\left(u^i_g\right) \le f\left(x^i_g\right) \\ x^i_g, & otherwise \end{cases} \tag{11}$$

3 THE EVSDE ALGORITHM

R. Storn [2] proposed a classical DE algorithm based on the DE/rand/1/bin. In the classical DE algorithm, the mutation factor and crossover rate are respectively 0.5 and 0.9 which are called experience parameters. Since the control parameters are invariant in progress of the evolution, the classical DE algorithm is not good enough to meet the various stages of evolution. C Chang [6] proposed a new adaptive DE algorithm which adapted the control parameters F and CR associated with each individual. J. Zhang [4] proposed an adaptive DE with optional external archive which implements a mutation strategy "DE/current-to-pbest" with optional archive and controls F and CR in an adaptive manner [11]. The experimental results have shown that the robustness of the DE algorithm and convergence rate would be enhanced if adaptive or self-adaptive parameter control is well designed.

To improve the existing adaptive and self-adaptive differential evolution algorithm, a new self-adaptive differential evolution algorithm EVSDE is raised here. The mutation factor F in the EVSDE algorithm is considered as a variable of a component, so that the dimension of variable $x^i_{j,g}$ is extended from D to D+1. Firstly, the mutation factor F is initialized randomly and applied to the progress of the mutation operator. In course of EVSDE algorithm running, F goes also through the progresses of mutation, crossover and selection as the variable $x^i_{j,g}$. After producing a new generation of variable, a new mutation factor is acquired. If the new generation of objective function value is better than

father generation, the new mutation factor will replace the father generation of one. With the evolution, the most appropriate mutation factor value can be gotten in the self-adaptive method.

The EVSDE algorithm mainly implements two specific improvements as follows on the basis of classical DE algorithm.

3.1 Proposing A New Mutation Strategy

There are many different mutation strategies used to possess different capabilities in various search phases of the evolution process. In Section 2.3, this paper have induced several strategies, and many literatures have already studied on the mutation strategies, for example, Q. Pan [12] proposed the SSPDE algorithm. Inspired by the literature [11], this paper proposed a new mutation strategy and the working of new strategy is as follows:

$$v_g^i = \frac{(gen-g)}{gen} \cdot x_g^{r1} + F \cdot rand \cdot (x_g^{r2} - x_g^{r3}) \quad (12)$$

where v^i_g is a mutation vector according to the target vector x^i_g, *gen* is evolution generation index and *g* is current generation index. r_1, r_2 and r_3 is randomly chosen integer which is different from each other and the running index *i* simultaneously.

3.2 Self-adaptive Parameters Setting by Extension Variable Dimension

This paper regards the mutation parameter *F* as a variable of the component and expands the variable dimension.

The initial value of *F* is randomly produced according to the following formula

$$F_g = F_{\min} + rand \cdot (F_{\max} - F_{\min}) \quad (13)$$

where *g*=0, $F_{\max}$ and $F_{\min}$ to denote the greatest and least value of *F*, respectively.

Then the value of *F* will serve as a variable of the component and go through mutation, crossover and selection, and produce a new generation *F*. If the next generation of the solution function fitness value is better than the current generation *F*, the new *F* will replace the last generation of *F*. The crossover rate *CR* adopts the linear diminishing strategy [6] defined as formula (14).

$$CR_g = CR_{\max} - \frac{g \cdot (CR_{\max} - CR_{\min})}{gen} \quad (14)$$

On the basis of the experience value, the control parameters follow the individual variable and implement the dynamic self-adaptive process in the evolutionary process. It thus helps to improve the robustness of the algorithm and avoids premature convergence.

4 EXPERIMENTIAL SETTING AND NUMERIAL RESULTS

In this section, 10 well-known benchmark functions as shown in Table I are used to measure the performance of the proposed algorithm and compared with the classical DE algorithm [6] and adaptive DE algorithm (ADE). The kind of ADE algorithm adopts the same mutation strategy as this paper and the control parameters adopt the strategy of literature [6].

Table 1: Ten Test Functions of Dimension D

Test Function	Initial Range
$f_1(x)=\sum_{i=1}^{D} x_i^2$	$[-100,100]^D$
$f_2(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10,10]^D$
$f_3(x)=\sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-100,100]^D$
$f_4(x)=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	$[-30,30]^D$
$f_5(x)=\sum_{i=1}^{D}\lfloor x_i+0.5\rfloor^2$	$[-100,100]^D$
$f_6(x)=\sum_{i=1}^{D}-x_i\sin\sqrt{\lvert x_i\rvert}$	$[-500,500]^D$
$f_7(x)=\sum_{i=1}^{D} ix_i^4+rand[0,1)$	$[-1.28,1.28]^D$
$f_8(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	$[-5.12,5.12]^D$
$f_9(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	$[-600,600]^D$
$f_{10}(x)=4x_1^2-\sum_{i=1}^{D}2.1x_i^4+x_1^6/3+x_1x_2-4x_2^2+4x_2^4$	$[-5,5]^D$

The parameters used in all the three algorithms are designated as follows: the population size is taken as 100, and the evolution generation index is fixed at 1000. The dimension of all test functions is fixed at 30 except the 10th function. The dimension of the 10th function is 2.

For the classical DE algorithm, the mutation factor *F* and crossover rate *CR* is 0.5 and 0.9 respectively. Due to the control parameters linearly changed, the mutation factor and crossover rate are respectively chosen to 0.1 and 0.9 in ADE algorithm. In EVSDE algorithm, the lower and upper bounds of *F* and *CR* are both in the range of [0, 1]. The experiment is running for 10 times and the average minimum value of 10 times and variances are calculated. The results are shown in Table 2. In the table, the best ones are highlighted with bold letters.

It can be seen that the optimal values obtained by the EVSDE algorithm are much better than the ones obtained by the classical DE algorithm in the most of functions, while the optimal values obtained by EVSDE algorithm searched are the closest to the theoretical optimal values. The experimental results show that EVSDE algorithm is superior to other two kinds of DE algorithm in accuracy and convergence speed for the most functions.

Table 2: Results of 10-Dimensional Problems

Function	$f_{\min}$	DE/rand/1 Mean/Std Dev	ADE Mean/Std Dev	EVSDE Mean/Std Dev

f_1	0	1.1277e-046/1.0496e-047	4.2389e-192/4.2389e-193	**0/0**
f_2	0	3.1453e-026/1.2719e-027	2.4388e-102/2.4206e-103	**2.5969e-248/2.4000e-249**
f_3	0	1.1882e+004/1.459e+003	1.3514e+004/1.1458e+003	**1.3323e-171/1.1723e-172**
f_4	0	4.7485/0.0190	8.5781/0.0607	**0.9688/0.0154**
f_5	0	**0/0**	**0/0**	**0/0**
f_6	-12569.5	-1.4922e+003/10.1448	-1.5447e+003/34.4507	**-2.9938e+003/1.6949**
f_7	0	**0/0**	**0/0**	**0/0**
f_8	0	1.5363e-005/3.7673e-005	2.3436e-017/1.4537e-018	**2.5643e-077/4.5643e-078**
f_9	0	2.0434e-007/2.0432e-008	1.7540e-007/1.3997e-007	**0/0**
f_{10}	1.0316285	2.7975/4.4409e-017	2.6593/0.0135	**1.7972/2.6079e-025**

5 CONLUSION

In this paper, a new self-adaptive DE based on extension variable dimension has been presented, and a new mutation strategy is proposed at the same time. The incorporating adaptation of control parameters and the new mutation strategy increase the convergence rate but also maintain the reliability of the EVSDE algorithm. It can be seen from the experimental results of 10 benchmark functions that the proposed algorithm is more efficient than the other two kinds of DE algorithms.

6 REFERENCES

[1] M. M. Ali, Differential evolution with generalized differentials, *Journal of Computational and Applied Mathematics*, 2010(10): 256-268, 2010.

[2] Ayed Salman, Andries P, Engelbrecht, Mahamed G.H.Omran, Empirical analysis of self-adaptive differential evolution, *European Journal of Operational Research,* 2007(183): 785-804, 2007.

[3] Storn R, Price K, Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces technical report, *International Computer Science Institute*, 1995(18): 22-25, 1995.

[4] Jingqiao Zhang, Arthur C. Sanderson, JADE: adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary,* 13(5): 945-958, 2009.

[5] Storn R, On the usage of differential evolution for function optimization, *Annual Conference of the North American Fuzzy Information Processing Society,* 1996: 519-523.

[6] Chang C S, Xu D Y, Differential evolution based tuning of fuzzy automatic train operation for mass rapid transit system, *IEEE Proceedings Electric Power Applications,* 147(3): 206-212, 2000.

[7] J. Teo, Exploring dynamic self-adaptive populations in differential evolution, *Soft Computing*, 2006(10): 673–686, 2006.

[8] Radha Thangaraj, Millie Pant, Ajith Abraham, New mutation schemes for differential evolution algorithm and their application to the optimization of directional over-current relay settings, *Applied Mathematics and Computation,* 2010(216): 532-544, 2010.

[9] Bo Liu, Hannan Ma, Xuejun Zhang, Yan Zhou, A memetic co-evolutionary differential evolution algorithm for constrained optimization, *Evolutionary Computation*, 2007: 2996-3002.

[10] R, Storn, System design by constraint adaptation and differential evolution, *IEEE Transactions on Evolutionary Computation,* 1999(3): 22–34, 1999.

[11] Wenjing Jin, An improved self-adapting differential evolution algorithm, *2010 International Conference on Computer Design and Appliations,* 2010: 341-345.

[12] Quan-Ke Pan, P.N. Suganthan, Ling Wang, Liang Gao, R. Mallipeddi, A differential evolution algorithm with self-adapting strategy and control parameters, *Computers & Operations Research*, 2011(38): 394-408.