International Journal of Hybrid Information Technology
Vol. 9, No.12 (2016), pp. 209-220
http://dx.doi.org/10.14257/ijhit.2016.9.12.19

A Novel Self-Learning Differential Evolution Algorithm in Two-State Dynamic Optimization

Feng Guiliang[1,2,3], CaoNing[1,2,3] and Zhang Xiao[1,2,3,*]

[1]*School of Information Science and Engineering, Hebei North University,China*，
[2] *Population Health Informazation in Hebei Province Engineering Technology Research Center, China* ，
[3] *Medical Informatics in Hebei Universities Application Technology Research and Development Center, China*
6838710@qq.com ,8018997@qq.com, xz1965cn@aliyun.com

Abstract

In this paper we propose a novel differential evolution algorithm based on self-learning, in order to improve the environment adaptive ability of the population in dynamic optimization. The proposed algorithm can monitor the environment changes using re-evaluation of individuals. We direct the population evolution based on the current best individual and another two random individuals, so that the convergence speed is faster and the diversity of the population is maintained. In this way we may reduce the influence from the frequent environment changes. Testing on six dynamic functions, we study the influences caused by period and dimensions. We also compared the proposed algorithm with existing algorithms, the experimental results show that our algorithm has a better environment adaptive ability and achieves better optimization result.

Keywords*: Intelligent computing; Differential evaluation; Dynamic optimization; Self-learning*

1. Introduction

In the large amount of engineering and science optimization problems, the variables, target functions, and constraints are generally changing through time. The topological structure of the function is changing. Therefore, the optimization result achieved is only valid in a short time duration. The optimization of such problem becomes very complicated. Currently, the dynamic environment can be categorized into three categories[1]. (1) In the first category, the constraints are changing, such as the Dynamic Knapsack Problem. (2) In the second category, the dynamic testing environment is caused by exclusive-or operation based on binary coding. The exclusive-or operator is performed on the binary coding in the individuals and causing the environment to change. (3) The third one is the changing of topological structure of functions, such as the dimension, height, width, and location of peak. In the above mentioned dynamic optimization problems, the last one is difficult to solve. Multiple factors are changing at the same time and cause the complexity of the problem to increase. The optimization result is usually not satisfactory. The computational intelligent algorithms have shown their ability in handling such dynamic optimization problems[2]. However, the history information is not always used due to the changing of the environment, and this reduces the algorithm efficiency. During the fast changes of the environment, the population has to adaptive to the new environment in time, and this is a great challenge to the existing learning algorithms. When the population are trapped in local extremes, all individuals

ISSN: 1738-9968 IJHIT

converge towards the same direction and the population lose the ability to evolve. In order to better track the changes of the extreme points, the population must keep a good diversity level. Cobb et al.[3] uses super mutation and local search to increase the diversity. When the change of environment is detected, the mutation rate is increased instantly or gradually. Yang et al. [4] propose an incremental learning algorithm based on population, and individual learning ability is enhanced. Yang et al.[5] propose a new method of directive individual updating. Fernandes et al. [6] increase the individual crossover based on Hamming distance. Better diversity is achieved on the population with further Hamming distance. In reference [7] and [8], prediction and memory is introduced into dynamic evolution algorithms. Prediction model is used when the environment factors are changing. Dong et al. [9] Propose an improved algorithm based on the Oracle penalty function and adaptive differential evolution. Their algorithm may reduce the number of parameters. Reference [11] uses a method based on differential evolution and adaptive constraint handling to track the dynamic changes in the environment. Thanh et al. [10] give a good review of swarm intelligence and dynamic optimization, and analyze the different performances of optimization algorithms.

In order to reduce the changes in the environment and reduce the complexity of the optimization problem, we propose a novel self-learning differential evolution algorithm (SeDE, Self-Learning Differential Evolution). Our proposed algorithm may keep the population diversity, enhance the influence of elite individuals, and make use of the history information to improve the algorithm efficiency.

2. Dynamic Function Design

Dynamic Optimization Problems (DOPs) can be represented as：

$$\min f(x,t) \quad s.t. \quad \begin{cases} h_i(x,t)=0, i=1,2\cdots,m \\ g_j(x,t)<0, j=1,2,\cdots,n \end{cases} \tag{1}$$

where $f(x,t)$ is the target function with respective to time, $h_i(x,t)=0$ is the i-th equality constraint condition at time t. There are m constraints in total. $g_j(x,t)<0$ is the j-th inequality constraint condition related to time t, there are n constraints in total.

When time t is changing, the dimension, the extreme value, and the location of the extreme of the function $f(x,t)$ may all change. In this paper we study the dynamic optimization problem caused by the change of the location of extremes. Given the n dimensional function f(x) under static environment, the i-th state is $o_i(c_{i1},c_{i2},\cdots,c_{in})$, $i=1,2,\cdots,K$, and the dynamic function is:

$$F(x,o,t)=f(\varphi(x,o),t) \tag{2}$$

where $F(x,o,t)$ is the dynamic function of time, $\varphi(x,o)$ is the mapping between variable x and state o, t is the time variable in f(x), such as iteration number, physical time, *et al.*

Use two dimensional Sphere function $f(x_1,x_2)=x_1^2+x_2^2$ as an example, given two points $o_1(c_{11},c_{12})$ and $o_2(c_{21},c_{22})$, the dynamic function is:

$$F(x,o,t)=(x_1-o_{i1})^2+(x_2-o_{i2})^2 \tag{3}$$

when t satisfies condition τ, i=1, otherwise, i=2. τ maybe the control parameter related to the generation number. When t changes, the minimal of the function is o_1 or o_2.

All printed material, including text, illustrations, and charts, must be kept within the parameters of the 8 15/16-inch (53.75 picas) column length and 5 15/16-inch (36 picas) column width. Please do not write or print outside of the column parameters.

Margins are 1 5/16 of an inch on the sides (8 picas), 7/8 of an inch on the top (5.5 picas), and 1 3/16 of an inch on the bottom (7 picas).

3. Adaptive Differential Evaluation Algorithm

SeDE algorithm is based on the dynamic detection of the environment. The algorithm is consisted of three parts: the dynamic environment detection, the individual two stage learning, and parameter adaptation. As shown in Table 1.

Table 1. SeDE Algorithm Framework

Alg 1 : SeDE Input : the target function $f(x)$ and domain of definition; Output: the optimized fitness of function $f(x)$
Step 1: Initialize population P: Initialize population P in the domain of definition, NP individual, D dimension, $\mathrm{P}=\{x_{ij}\}, i=1,\cdots,NP, j=1,\cdots,D$. Initial parameter F and CR.
Step 2: Dynamic environment detection: detect the environment changes if the environment changes go to Step 3, otherwise go to Step 4.
Step 3: Learning stage 1: Get the state of the current environment, and use the current best solution to direct the population P.
Step 4: Learning stage 2: direct population P based on the current best individual.
Step 5: Evaluate population P, select the outstanding individuals from parent and the offspring generation.
Step 6: Adjust the control parameters: use the adaptive method to update the mutation step F and crossover rate CR.
Step 7: Record the optimal solution x^* and the related fitness value $fit = f(x^*)$
Step 8: When termination criteria is met, output statics, otherwise go to Step 2.

3.1. Environment Detection Method

The environment detection method includes two stages: check if it has been changed and get the state of the environment. The character of the environment determines the method of detection [12], in this paper, our dynamic optimization problem has two different states. The individual fitness changes significantly when the state transfers from one to the other. In Figure 1, we use the minimization problem of the two-dimensional Sphere function $f(x_1, x_2) = x_1^2 + x_2^2$ as an example.

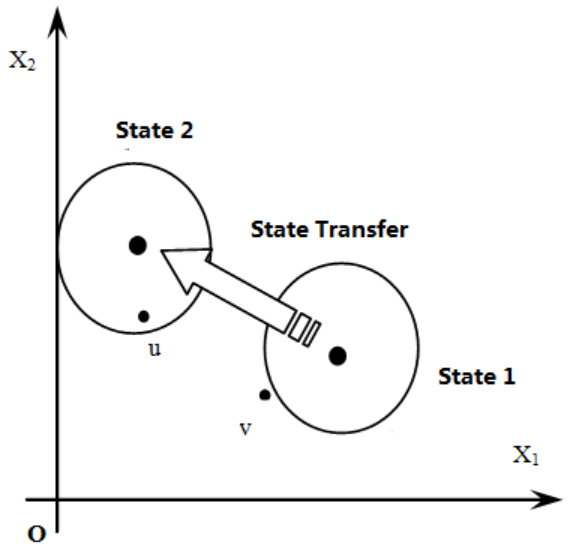

Figure 1. The Fitness Value Change Caused By the Environment Change

The two states are represented as o_1 and o_2. u and v are the random individuals in the population. When the optimization environment is o_2, individual u is close to the state center and better that individual v. When state transfers from o_2 to o_1, the fitness of individual u decreases significantly and worse than that of individual v. Therefore, in this paper, we use the specific individual as a monitor of the environment change. For instance, we may choose the best individual in the population or the random individual in the solution space.

The second step is to determine the state of the environment. We use the variable $changTime$ to calculate the number of times the environment changed. Initial value is set as: $changTime = 1$. When environment changes, update the variable as: $changTime = changTime + 1$. We use the logical variable $Status$ to denote the current environment state: $Status = \mathrm{mod}(changeTime, 2)$, where mod denotes the mode 2 operation. The environment is at state 1 when $Status = 1$, otherwise the environment is at state 2.

3.2. Individual Self-Learning Method

To be better adapted to the changing environment, we use the elite individuals to lead the learning process. Based on the space-time location of the evaluation, the learning process is divided into two: in the first stage, the population learn from the best solution in the history under state j, when the environment transfers from state i to state j : $i \neq j, i, j \in \{1,2\}$. In the second stage, the population learn from the current best individual. Detailed algorithm is given in Table 2.

The main title (on the first page) should begin 1 3/16 inches (7 picas) from the top edge of the page, centered, and in Times New Roman 14-point, boldface type. Capitalize the first letter of nouns, pronouns, verbs, adjectives, and adverbs; do not capitalize articles, coordinate conjunctions, or prepositions (unless the title begins with such a word). Please initially capitalize only the first word in other titles, including section titles and first, second, and third-order headings (for example, “Titles and headings” — as in these guidelines). Leave two blank lines after the title.

Table 2. Individual Self-Learning

Alg 2 : Self-learning algorithm
Input : Population – P,
fitness value – fit,
best solution in histoy – stageBestIndi,
corresponding fitness –stageBestFit,
environment change counter –changeTime=1.
Output : Environment vector - v

Step1: Get the current best solution: bestIndi, and its fitness value: bestFit

Step2: if bestFit ≠ Revaluate(bestIndi), re-evaluate the optimal solution and determine whether the environment has changed.

Step3: flag=mod(changeTime, 2)

Step4: if flag==2, State 1 transfers to state 2.

Step5: if bestFit<stageBest(1), Update the history best solution and fitness value of state 1:

Step6: update(bestIndi, bestFit) ;

Step7: for all individual x in population p, Learn from the best individual in history of state 2: v=x+w*(stageBestIndi(2)-x) .

Step8: else transfer from state 2 to state 1.

Step9: if bestFit<stageBest(2), update the best solution in history and its fitness value of state 2:

Step10: update(bestIndi, bestFit).

Step11: for all individual x in population P, learn from the best individual of history of state 1: v= w*(stageBestIndi(1)-x)

Step12 changeTime=changeTime+1;

Step13: else learn from the current best individual:

Step14: for all individual x in population P:

$v_i = bestIndi + F*(bestIndi - randP_1) + k*(randP_2 - randP_3)$

Learn from the best solution in history: When the environment state transfers from i to j, the fitness value and the individual ranking are both changed. The population need to be adapted to the environment as soon as possible. We use the best solution in history of sate j as the template for other individuals to learn.

Given the best solution in history of j: $stageBest(j)$, we have the learning strategy when state transfers from i to j:

$$x\_new = x + \varpi * (stageBest(j) - x) \quad (4)$$

where, x_new is the new individual based on x and ϖ is the disturbance parameter.

In Figure 2 we give the example of two-dimensional and two-state function Sphere to explain the learning process of individual x. The individual under state 1 is directed to point v by the best solution of history of sate 2. It is closer to the optimal solution under state 2 and has a better fitness value.

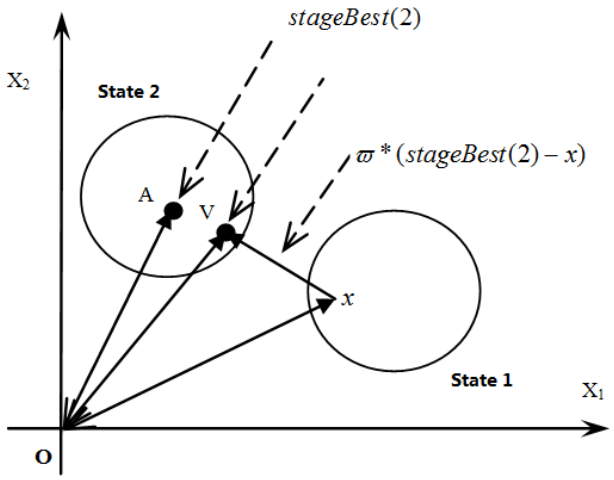

Figure 2. Individual Self-Learning Method

Learn from the current best solution: the evaluation times, iteration numbers may be treated as the driven force of the population evolution. In the dynamic optimization problem, due to the environment change the algorithm needs to fit the new environment in time. Therefore, we direct the individuals to learn from the current best solution:

$v_i = bestIndi + F*(bestIndi - randP_1) + k*(randP_2 - randP_3)$, where, v_i is the over test vector of the i-th individual, $bestIndi$ is the best individual of the current generation, and $randP_j, j = 1,2,3$ is the individual selected from population P that is different from $bestIndi$ or current individual. F is the control parameter of mutation step, and k is the even distribution in domain (0,1].

3.3. Individual Crossover and Updating

Crossover on v_i and P_i and generate target vector u_i: $u_i = (u_{i1}, u_{i2}, \cdots, u_{iD})$. We have:

$$u_{ij} = \begin{cases} v_{ij} & if\ U(0,1) < CR \text{ or } j = j\_rand \\ P_{ij} & \text{otherwise} \end{cases} \quad (5)$$

where, $U(0,1)$ is the random distribution in domain [0,1], CR is the crossover rate, and j_rand is the random integer in $[1,\cdots,D]$. Ensure there is at least one different dimension between u_i and x_i. Select the better individual $x_i^{'}$ from u_i and x_i for the next generation.

$$x_i^{'} = \begin{cases} u_i & \text{if } u_i \text{ is superior } x_i \\ x_i & \text{otherwise} \end{cases} \quad (6)$$

3.4. Parameter Adaptation

Individuals in population P, are corresponded to mutation step F and crossover rate CR, all of them evolve at the same pace. In g+1 generation, we update the F and CR of the i-th individual as follows [12]:

$$F_{i,g+1} = \begin{cases} F_l + rand1 * F_u & \text{if } rand2 < \tau_1 \\ F_{i,g} & \text{otherwise} \end{cases} \quad (7)$$

$$CR_{i,g+1} = \begin{cases} rand3 & \text{if } rand4 < \tau_2 \\ CR_{i,g} & \text{otherwise} \end{cases} \quad (8)$$

where, $rand_j$ j=1,2,3,4 is the random number in $[0,1]$, τ_1 and τ_2 are the adjustment probability, and they are both set to 0.1. We also set: F_l =0.1, $F_u = 0.9$

4. Experimental Results

To verify the algorithm performance, we select six benchmark functions, as shown in Table 3. We use different parameter configurations and compared the proposed algorithm with basic differential evaluation algorithm and adaptive differential evolution algorithm. The selected static functions include unimodal functions, f_1 to f_4, and multimodal functions, f_5 and f_6. In order to simulate the dynamic environment, we randomly generate fix point O in the domain of definition of function $f(x)$. Let $f^*(x) = f(x-o)$, and the states transfer between $f(x)$ and $f^*(x)$.

Table 3. The Testing Functions

Function	State Space
$f_1(x) = \sum_{i=1}^{D} x_i^2$	$-10 \le x_i \le 10$
$f_2(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$-100 \le x_i \le 100$
$f_3(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le D \}$	$-100 \le x_i \le 100$
$f_4(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$	$-30 \le x_i \le 30$
$f_5(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i) + 20 + e$	$-32 \le x_i \le 32$
$f_6(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$-600 \le x \le 600$

4.1. Algorithm Performance under Low Dimension and Dynamic Environment

In order to test the SeDE algorithm in low dimension optimization problem, we test the algorithm independently for 25 times and each time we re-initialize the population in the domain of definition. The population size is set to 30 and the iteration maximum time is set to 2500. The frequency of environment change is set to 25 generations. The optimization accuracy is set to 1E-8, and the simulation results are shown in Table 4.

Table 4. Optimization Results of SeDe Algorithm under Low Dimension Dynamic Environment

Function	D=5 Mean(Std)		D=10 Mean(Std)	
	State 1	State 2	State 1	State 2
F1	3.31E-03 (6.60E+00)	4.76E-04 (2.76E-01)	1.20E-01 (1.41E+02)	4.59E-02 (2.50E+01)
F2	9.31E-03 (1.90E-01)	3.57E-03 (1.87E-02)	8.05E-02 (1.09E+00)	4.97E-02 (4.81E-01)
F3	3.51E-05 (6.93E-03)	2.85E-06 (3.99E-03)	1.77E-06 (2.92E-03)	2.51E-07 (7.26E-04)
F4	9.52E-07 (6.20E+01)	7.57E-05 (2.83E+01)	5.57E-04 (2.19E+03)	7.56E-04 (1.21E+03)
F5	3.55E-02 (6.69E-01)	1.50E-02 (1.23E-01)	2.12E-01 (1.66E+00)	1.59E-01 (8.95E-01)
F6	1.77E-01 (1.31E-01)	1.05E-01 (7.89E-02)	5.68E-01 (2.04E+00)	5.44E-01 (2.57E-01)
Score	3.75E-02	2.07E-02	1.64E-01	1.33E-01

In the experiment we record the minimum value $pbest(k,i,j)$ in the j-th period of i-th independent test under k-th state. To get the optimal value $sbest(k)$ we run the test for 25 times and 1E+5 generations. As shown in Table 4, the mean and standard deviation of the optimal values are calculated. We can see that when dimension equals 5, the score is similar to each other. When the dimension increases to 10, the scores are significantly different. When the dimension increase from 5 to 10, the problem space increases, and the complexity increases. The optimization results therefore become worse, the changing environment is more difficult to chase.

When the environment state transfers from i-th state to j-th state, the individual patent under i-th state is destroyed. The individual learning process under i-th state is also a destruction to the patent under j-th state. This repeated process causes quasi-periodic change in the fitness curve, as shown in Figure 3.

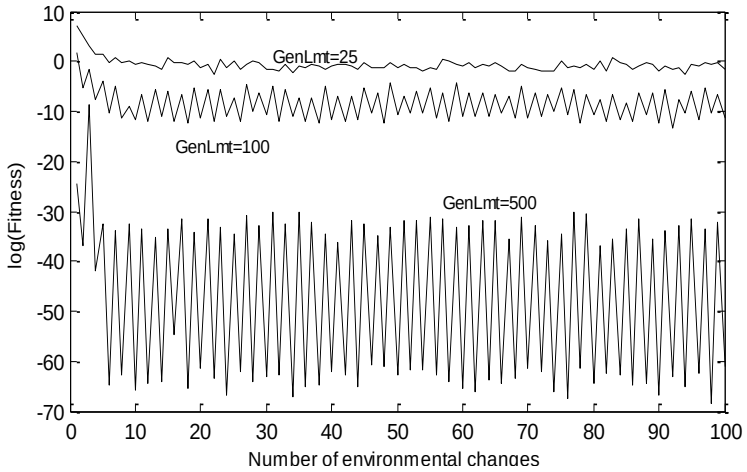

Figure 3. Fitness Curve of the Population in Dynamic Environment

4.2. Comparison of Algorithms Performances under High Dimension and Dynamic Environment

In this experiment, we compared the influence of different environment change frequencies. We compared our algorithm with the standard differential evolution algorithm and the adaptive differential algorithm in reference [12]. The algorithm is tested independently for 25 times, the mean of optimal value is used as a static index. In order to maintain the same number of changes under different environment changing period, we set the iteration number as GenLmt*100. For all three algorithms, the population size is set to 30 and the dimension is set to 30, initialization is performed randomly in the solution space. The simulation results are shown in Table 5 and Table 6. Compared with other two algorithms, the SeDE algorithm has a obvious advantage. When genLmt equals to 5 or 10, the environment is changing quickly and the individual has to be adapted to search for the optimal solution. This requires the algorithm to have a faster learning speed. In the SeDE algorithm, the best individual in history will lead the individual learning under the environment change, and the current best solution will lead the learning afterwards. This character gives the SeDE algorithm a better learning ability, and may be better adapted to new environment than the other two algorithms. When genLmt equals to50 and 100, the change of environment slows down. This requires the algorithm to maintain population diversity for multimodal function and avoid local extremes. The SeDE algorithm is based on DE algorithm which has a strong ability in keeping population diversity. The random individuals are included in the individual learning process to achieve better optimization results.

Table 5. Comparison of Algorithm Performance under Different Environment Change Frequency (A)

Function (30 dimension)	MeanBest(GenLmt=5)			MeanBest(GenLmt=10)		
	SeDE	DE	Reference [12]	SeDE	DE	Reference [12]
F1	3.11e+3	6.40e+3	4.27e+3	1.27e+3	3.03e+3	1.64e+3
F2	2.97e+9	3.24e+6	5.64e+7	3.56e+3	4.16e+6	9.61e+6
F3	2.10e-3	9.11e-3	3.79e-3	1.48e-3	3.17e-3	2.59e-3
F4	6.10e+4	1.66e+5	8.41e+4	2.31e+4	9.03e+4	4.30e+4
F5	1.21e+1	9.65e+0	7.07e+0	6.93e+0	7.40e+0	4.55e+0
F6	2.80e+1	6.02e+1	3.23e+1	1.71e+1	2.26e+1	1.74e+1
Score	4	1	1	5	0	1

Table 6. Comparison of Algorithm Performance under Different Environment Change Frequency (B)

Function (30 dimension)	MeanBest(GenLmt=50)			MeanBest(GenLmt=100)		
	SeDE	DE	Reference [12]	SeDE	DE	Reference [12]
F1	2.13e+2	4.74e+2	2.40e+2	3.22e+1	1.34e+2	5.34e+1
F2	1.95e+0	9.38e+0	3.81e+0	7.38e-1	2.91e+0	2.00e+0
F3	4.21e-4	8.13e-4	5.55e-4	1.14e-4	1.36e-4	1.52e-4
F4	6.35e+3	2.42e+4	5.78e+3	1.57e+3	1.58e+4	8.17e+2
F5	1.70e+0	4.13e+0	2.53e+0	6.48e-1	3.05e+0	2.32e+0

F6	2.79e+0	5.63e+0	3.17e+0	1.16e+0	2.45e+0	1.51e+0
Score	5	0	1	5	0	1

5. Conclusion

In this paper we propose a novel algorithm SeDE based on the learning process directed by elite individuals. The new algorithm has the ability of DE algorithm to maintain population diversity. The environment changes is monitored by re-evaluation of specific individual, and the self-learning method is adopted to better fit the environment. When the environment is changed, the best individual in the history is adopted to direct the learning of the population at once, and the current best individual is adopted after the population enters the new environment. The diversity is maintained and the convergence speed is improved. The two-state environment change is taken as an example, and experimental results show that the proposed algorithm has a strong global and local search ability. However, in real world applications there are many multi-state problems to be solved, we will extend our research to this type of optimization problems in the future.

References

[1] J. R. Gonza´lez, D. A. Pelta and C. Cruz, "Optimization in dynamic environments: a survey on problems, methods and measures", Soft Computing, vol. 15, no. 1, **(2011)**, pp. 1427-1448.
[2] S. Yang and C. Li, "A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments", IEEE Transaction. on Evolutionary Computation, vol. 14, no. 6, **(2010**), pp. 959-974.
[3] H. G. Cobb and J. J. Grefenstette, "Genetic Algorithms for Tracking Changing Environments", International Conference on Genetic Algorithms, Morgan Kaufmann, **(1993)**, pp. 523-530.
[4] X. Yao and S. Yang, "Population-based incremental learning with associative memory for dynamic environments", IEEE Transactions on Evolutionary Computation, vol. 12, no. 5, **(2008)**, pp. 542- 561.
[5] S.Yang, "Genetic algorithms with memory and elitismbased immigrants in dynamic environments", Evolutionary Computation, vol. 16, no. 3, **(2008)**, pp. 385-416.
[6] A. C. Rosa and C. M. Fernandes, "Self-adjusting the intensity of assortative mating in genetic algorithms", Soft Computing, vol. 12, no. 10, **(2008)**, pp. 955-979.
[7] H. Chen, M. Li and X. Chen, "Multi-population Evolutionary Algorithm with Forecast Scheme in Dynamic Environment", Journal of Chinese Computer Systems, vol. 33, no. 4, **(2012)**, pp. 796-781
[8] H. Chen，L. Ming and X. Chen, "Hybrid memory scheme for genetic algorithm in dynamic environments", Journal of Applied Science, vol. 28, no. 5, **(2010)**, pp. 540-545.
[9] M. Dong, X.i Cheng and Q. Niu, "Adaptive constrained differential evolution algorithms based on Oracle penalty function", Computer Applications and Software, vol. 31, no. 1, **(2014)**, pp. 290-294
[10] N.T. Thanh, S. Yang and J. Branke, "Evolutionary Dynamic Optimization: A Survey of the State of the Art", Swarm and Evolutionary Computation, vol. 6, no. 10, **(2012)**, pp. 1-24.
[11] E. K. Silva, H. J. C. Barbosa and A. C. C. Lemonge, "An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization", Optimization and Engineering, vol. 12, no. 1, **(2011)**2, pp. 31-54.
[12] J. Brest, S. Greiner and B. Bošković, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems", IEEE Transactions on Evolutionary Computation, vol. 10, no. 6, **(2006)**, pp. 646 – 657

Authors

Feng Guiliang, he is a Lecturer of department of information science and engineering, Hebei North University, He is good at the field of software engineering and multimedia development.

Cao Ning, he is a Lecturer of department of information science and engineering,Hebei North University,He is good at the field of software engineering and multimedia development.

Zhang Xiao, he is a professor at the school of information science and engineering; Hebei North University, He is mainly in the field of medical informatics, software engineering.