A Linear map-based mutation scheme for real coded genetic algorithms

Yue-jiao Gong, , *Student Member, IEEE*, Xiao-min Hu, *Student Member, IEEE*,

Jun Zhang, *Senior Member, IEEE*, Ou Liu and Hai-lin Liu

***Abstract*—Real coded genetic algorithms (RCGAs) have been widely studied and applied to deal with continuous optimization problems for years. However, how to improve the degree of accuracy so as to produce high quality solutions is still one of the main difficulties that RCGAs face with. This paper proposes a novel mutation scheme for RCGAs. The mutation operator is defined as a linear map in the space of chromosomes (in RCGAs each chromosome is a floating point vector). It operates on a whole chromosome instead of several single genes to produce the new chromosome. The linear map is represented by a randomly generated mapping matrix which satisfies some predefined constraints. By this way, the constraints restrict the mutations of genes on a same chromosome as a whole. RCGA with the proposed mutation scheme is tested on 16 benchmark functions. Results demonstrate that the proposed scheme not only improves the solution accuracy that RCGA can obtain, but also presents a very fast convergence speed. The linear map-based mutation scheme has a bright future to improve RCGAs.**

I. Introduction

Genetic algorithms (GAs) [1][2], which were firstly proposed by Holland, have been successfully used for solving various optimization problems for years. Inspired by the natural genetic theory, a GA maintains a population of chromosomes which represent the candidate solutions for the problem and evolve through genetic operators generation by generation. Each chromosome in the population is associated with a fitness value which determines its possibilities for surviving in the next generation. In the selection, chromosomes with relatively high fitness values are more likely to be preserved or even duplicated, whereas chromosomes with low fitness values tend to be discarded. New chromosomes are generated by the crossover and mutation operators and some of them may have higher fitness values. Through the above evolutionary mechanism, GAs are capable of exploiting and exploring the initially unknown search space and bias the subsequent search into appropriate subspaces. GAs are good at searching in large, complex and hardly known problem spaces, where the traditional algorithms (enumerative, heuristic, etc.) have difficulties to deal with [3].

This work was supported in part by the NSFC Joint Fund with Guangdong under Key Project U0835002 and the National High Technology Research and Development Program ("863"Program) of China, No.2009AA01Z208.

Yue-jiao Gong, Xiao-min Hu and Jun Zhang are with the Department of Computer Science, Sun Yat-sen University, Guangzhou, China. Jun Zhang is the corresponding author (email: junzhang@ieee.org)

Ou Liu is with The Hong Kong Polytechnic University, Hung Hom, Hong Kong.

Hai-lin Liu is with Guangdong University of Technology,Guangzhou, China

In terms of real world applications, GAs using a real number representation scheme are commonly referred to as the real coded genetic algorithms (RCGAs). RCGAs have been widely studied [3][4], improved [5][6][7], and applied [8][9][10] in the literature. In RCGAs, each chromosome is a vector of floating point numbers. Each number corresponds to a variable in the problem to be optimized. The length of the vector is the same as the number of variables. Therefore, by using such an encoding scheme, each chromosome is directly a candidate solution of the problem. The real encoding scheme is more convenient for the problems with a large number of variables (which are difficult for the binary encoding scheme) and it is possible for achieving solutions with high precision [3].

However, how to improve the solution accuracy of RCGAs is still one of the main difficulties that RCGAs face with. This problem becomes quite serious when the problem space is high-dimensional and multimodal where vast local optima exist. For years, researchers have made great efforts to improve the RCGAs and proposed various modified versions. Using new mutation operators instead of the classical random mutation is a promising approach. Michalewicz [11] proposed a non-uniform mutation that used a function to generate new genes in a range which was initially large and shrunk in every generation. Li and Zhang [12] proposed a complex mutation strategy based on schema mutation. The mutation operation could protect the existing excellent individuals and promoted better individual production. Cui [13] proposed a Wavelet denoising mutation which fine-tuned the gene based on wavelet theory. Various adaptive mutation methods can be found in Greenwood [14], Peng *et al.* [15], Teo [16], Yuen and Chow [17], etc. These algorithms outperform the classical RCGA, but they are either too complex or not effective enough to improve the solution accuracy.

In order to overcome the above shortcomings, this paper proposes a novel mutation scheme based on a linear map. As mentioned before, in RCGAs each chromosome is a vector of floating point numbers. This encoding scheme enables us to define the mutation operator as a linear map in the space of floating point vectors. The linear map is represented by a randomly generated mapping matrix which satisfies some predefined constraints. The constraints restrict the mutation of the whole chromosome instead of the mutation of every single gene. (The restriction of the mutation of every single gene can be found in [11][18][19] in the literature.) By this way, the proposed mutation scheme has advantages of considering genes on a same chromosome as a whole. Then, in order to preserve the information of the original chromosome and treat every dimension without bias, in this paper we propose that the constraint of the randomly generated mapping matrix is that it should be orthogonal. Furthermore, we introduce the use of an orthogonal rotation-like mapping matrix. We test RCGA with the linear map-based mutation on 16 benchmark functions. Experimental results and comparisons with other algorithms demonstrate that the proposed mutation scheme contributes to high solution accuracy and fast convergence speed. Moreover, the scheme is easy to implement. The linear map-based mutation method has a bright future to improve the RCGAs.

The rest of this paper is organized as follows. Section II presents an introduction of linear map. Section III proposes and describes the linear map-based mutation scheme. In section IV, 16 benchmark functions are tested and experimental results are reported. At last, conclusions are drawn in Section V.

II. LINEAR MAP

This section presents an introduction of linear map. In mathematics, a linear map [20][21] (which is also called a linear transformation) is a mapping function defined between two vector spaces that preserves the vector addition and scalar multiplication (an operation that a real number times a vector to obtain a new vector). Linear maps from a vector space to itself are called endomorphisms.

Let X, Y be two vector spaces defined over a same field R, a mapping function $f: X \rightarrow Y$ is a linear map if any two vectors x_1, x_2 in X and any scalar a in R satisfy (1) and (2).

$$f(x_1 + x_2) = f(x_1) + f(x_2) \tag{1}$$

$$f(ax_1) = af(x_1) \tag{2}$$

In generalization, for any k vectors $x_1, x_2, \ldots, x_k$ in X and any k scalars $a_1, a_2, \ldots, a_k$ in R, equation (3) is also satisfied.

$$\begin{aligned} f(a_1x_1 + a_2x_2 + \cdots + a_kx_k) = \\ a_1f(x_1) + a_2f(x_2) + \cdots a_kf(x_k) \end{aligned} \tag{3}$$

If X and Y are finite-dimensional, in order to enable concrete calculations, we can define the linear map between X and Y as a matrix, i.e., if A is a $m \times n$ matrix, then the function $f(x) = Ax$ is a linear map $f: R^n \rightarrow R^m$.

Let $\{e_1, e_2, \ldots, e_n\}$ be the basis of X, and $\{v_1, v_2, \ldots, v_m\}$ be the basis of Y. So that every vector in X can be represented as $c_1e_1+c_2e_2+\ldots+c_ne_n$, which is uniquely determined by the coefficients $c_1, c_2, \ldots, c_n$. If $f: X \rightarrow Y$ is a linear map, then we have $f(c_1e_1+c_2e_2+\ldots+c_ne_n) = c_1f(e_1)+c_2f(e_2)+\cdots+c_nf(e_n)$, which infers that the mapping function f is entirely determined by the values of $f(e_j)$, j=1, …, n. Notice that $f(e_j)$ is a vector in Y, so that we have $f(e_j) = a_{1j}v_1+a_{2j}v_2+\ldots+a_{mj}v_m$, which infers that f is entirely determined by the values of a_{ij}. Therefore, through the mapping matrix $A = (a_{ij})_{m\times n}$, we can easily compute the $f(x)$ for any vectors x in X.

The linear map is a widely used mathematical operation defined on various kinds of vector spaces. In this paper, it is introduced to deal with the search space of RCGAs where the population of chromosomes evolves.

III. RCGA WITH LINEAR MAP-BASED MUTATION SCHEME

In this section, we first briefly introduce RCGAs and then propose and describe the linear map-based mutation scheme in detail.

Inspired by the natural genetic theory, GAs start with an initial population of chromosomes and evolve through the genetic operators in each generation. The chromosomes are the candidate solutions of the problem and composed of genes. RCGAs are the GAs using a real number representation scheme, in which each gene in a chromosome corresponds to a variable to be optimized in the problem.

The fundamental structure of GAs includes the evaluation of individual fitness, the selection to preserve and duplicate good individuals and discard bad individuals, and the recombination through crossover and mutation. An overall flowchart of GAs is shown in Fig. 1, in which the elitist strategy of preserving excellent individual is also included.

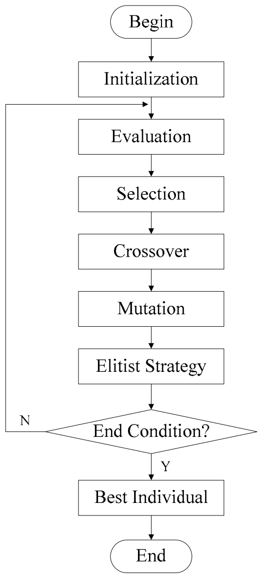

Fig. 1. Flowchart of GAs.

In the classical mutation scheme, new genes are uniform random numbers which are generated in the range $[a^i, b^i]$ (a^i and b^i are the lower bound and upper bound of the i^{th} variable). The effect of this scheme is equivalent to reinitializing the genes, which may destroy the already formulated population structure and interrupt the convergence process. In order to overcome this problem, some new mutation schemes have been proposed to generate new genes which are "close" to the original genes [11][18][19]. These mutation schemes outperform the classical mutation. However, some problems still exist. Firstly, the measure of the "close" is not definite. Secondly, all these schemes concentrate on a single dimension and neglect the interaction of different dimensions, i.e., genes on a same chromosome are not considered as a whole.

In this paper, a novel mutation scheme based on a linear map is proposed. By regarding each chromosome as a gene vector x, the search space as a vector space X, the mutation on x can be defined as $f(x)=Mx$, where M is a randomly generated mapping matrix which satisfies some constraints. By this way, the mutation concentrates on the whole chromosome instead of every single gene. We also modify the way of choosing the genes to be mutated as follows. For each chromosome (the gene vector), the mutation happens with a predefined mutation probability Pm, and two genes on the chromosome are randomly generated to be mutated.

Let $x_j = (x_j^1, x_j^2, \cdots, x_j^n)^{\mathrm{T}}$ be the jth gene vector, $j = 1, 2, \ldots, m$, where n is the number of variables to be optimized and m is the size of the population. Let $y_j = (y_j^1, y_j^2, \cdots, y_j^n)^{\mathrm{T}}$ be the vector of x_j after mutation. Therefore, $y_j = Mx_j$, where $M = (m_{ij})_{n\times n}$ is the mapping matrix. According to the definition in Section II, we have $y_j^i = m_{i1}x_j^1 + m_{i2}x_j^2 + \cdots + m_{in}x_j^n$. Notice that the genes on x_j are randomly chosen to be mutated, if x_j^i is not going to be mutated, we have $m_{ii} = 1$, $m_{ik} = m_{ki} = 0$, $k=1, \ldots, n$, $k \neq i$, i.e., $y_j^i = x_j^i$. For example, for a 4-dimensional gene vector x_j, if x_j^2 and x_j^3 are chosen to be mutated whereas x_j^1 and x_j^4 stay the same, the mapping matrix is similar to (4).

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_{22} & m_{23} & 0 \\ 0 & m_{32} & m_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Furthermore, we add several constraints to M so as to control the values of the genes to be mutated. In the following statement, in order to simplify the definition of the constraints, we extract a submatrix M_1 out of the mapping matrix M, which is composed of the mutated columns and rows of M, e.g., the submatrix of (4) is (5).

$$M_1 = \begin{bmatrix} m_{22} & m_{23} \\ m_{32} & m_{33} \end{bmatrix} \tag{5}$$

The constraints are defined as follows.

Firstly, in order to facilitate the analysis and calculation, M_1 should be non-degenerate.

Secondly, in order to preserve the information of the original gene vector and treat every dimension without bias, M_1 should be orthogonal, i.e., $M_1M_1^{\mathrm{T}} = I$.

Since finite-dimensional linear isometries such as rotations, reflections, and their combinations produce orthogonal matrices, in our experiments, we use the randomly generated 2D rotations as

$$M_1 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{6}$$

where θ is a random value uniformly distributed over $[-\pi, \pi]$.

The pseudo code for the mutation process is shown in Fig. 2, where c^i stands for the center of the ith dimension, $i = 1, 2, \ldots, n$, i.e., $c^i = (a^i+b^i)/2$. In order to make the rotation around the center vector $c = (c_1, c_2, \ldots, c_n)$ of the search space instead of the origin point $(0, 0, \ldots, 0)$, line (13) and (16) in Fig. 2 are necessary.

```
(1)  Procedure mutation
(2)    for each chromosome in the population
(3)      r = random_float(0,1);
(4)      if r > Pm
(5)        goto (2);
(6)      end if
(7)      else
(8)        k1 = random_int(1,n);
(9)        k2 = random_int(1,n);
(10)       if k2 = k1
(11)         goto (9);
(12)       end if
(13)       $x_j^{k_1} = x_j^{k_1} - c^{k_1}$ ; $x_j^{k_2} = x_j^{k_2} - c^{k_2}$ ;
(14)       θ = random_float(-π, π);
(15)       $(x_j^{k_1}, x_j^{k_2})' = M_1 \times (x_j^{k_1}, x_j^{k_2})'$ ;
(16)       $x_j^{k_1} = x_j^{k_1} + c^{k_1}$ ; $x_j^{k_2} = x_j^{k_2} + c^{k_2}$ ;
(17)         if $x_j^{k_1}$ or $x_j^{k_2}$ is out of range
(18)           goto (13)
(19)         end if
(20)     end if
(21)   end for
(22) end procedure
```

Fig. 2. Pseudo code for the mutation process.

We want to note that the constraints and the mapping matrix can be defined in various ways. In this paper, we only describe the linear map-based mutation method by using an orthogonal matrix as an example.

Experiments and discussions of RCGA with the proposed mutation scheme are presented in the next section.

IV. Experimental results

In this paper, a novel mutation scheme based on a linear map for RCGAs is proposed. In the following, RCGA with the proposed mutation scheme is abbreviated to LM_RCGA. In order to compare the performance of LM_RCGA with RCGA, 16 benchmark functions [22] are tested.

TABLE I
BENCHMARK FUNCTIONS

Function	n	Domain	Optimum
$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^n$	0
$f_2(x)=\sum_{i=1}^{n}\|x_i\|+\prod_{i=1}^{n}\|x_i\|$	30	$[-10,10]^n$	0
$f_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{n}x_j)^2$	30	$[-100,100]^n$	0
$f_4(x)=\max\{\|x_i\|,1\le i\le n\}$	30	$[-100,100]^n$	0
$f_5(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	$[-30,30]^n$	0
$f_6(x)=\sum_{i=1}^{n}(\|x_i+0.5\|)^2$	30	$[-100,100]^n$	0
$f_7(x)=\sum_{i=1}^{n}ix_i^4+random[0,1)$	30	$[-1.28,1.28]^n$	0
$f_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\|x_i\|}$	30	$[-500,500]^n$	-12569.5
$f_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	30	$[-5.12,5.12]^n$	0
$f_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i)+20+e$	30	$[-32,32]^n$	0
$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-600,600]^n$	0
$f_{12}(x)=\frac{\pi}{n}\{10\sin^2(\pi y_i)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4),$	30	$[-50,50]^n$	0
$f_{13}(x)=0.1\{\sin^2(3\pi x_1+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_n-1)[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50,50]^n$	0
$f_{14}(x)=-\sum_{i=1}^{5}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0,10]^n$	-10.1532
$f_{15}(x)=-\sum_{i=1}^{7}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0,10]^n$	-10.4029
$f_{16}(x)=-\sum_{i=1}^{10}[(x-a_i)(x-a_i)^T+c_i]^{-1}$	4	$[0,10]^n$	-10.5364

A. Experimental Setup

The 16 benchmark functions are shown in Table I, where f_1-f_{13} are high-dimensional functions and f_{14}-f_{16} are low-dimensional functions. f_1-f_5 are unimodal functions, f_6 is a step function, f_7 is a noisy quartic function, and f_8-f_{16} are multimodal functions, among which f_8-f_{13} have a great number of local optima. For example, Fig. 3 illustrates the local optima of the two dimensional f_9.

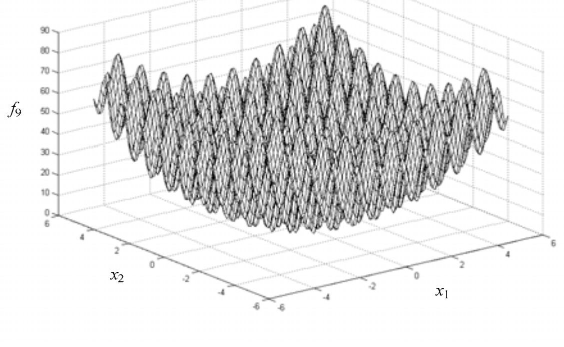

Fig. 3. The figure of two dimensional f_9.

In the experiments, the same population size *PopSize*=100, the same max number of generations *MaxGens*=10000, the same probability of crossover and mutation *Px*=0.2 and *Pm*=0.07, and the same tournament size for selection *TourSize*=10 are set to both RCGA and LM_RCGA. For each function, 100 independent trials are carried out by applying the RCGA and LM_RCGA under the same circumstances.

B. Experimental Results and Comparison

The comparisons of solution accuracy are shown in Table II, where the mean values, the best values and the standard deviations of the results found by RCGA and LM_RCGA over the 100 trials are listed. It can be seen that for 12 out of the 16 benchmark functions, LM_RCGA achieves better mean values. For 14 out of the 16 benchmark functions, LM_RCGA achieves better best values. And for 11 out of the 16 benchmark functions, LM_RCGA achieves better standard derivations. We also have applied a two-tailed test to judge the performance between RCGA and LM_RCGA. The t-test results show that for most of the benchmark functions, the enhancements by LM_RCGA are significant.

The convergence curves of the mean values during the training process in solving the f_1, f_6, f_7, f_9, f_{12}, and f_{14} are shown in Fig. 4, where the vertical axis is the mean function value of

TABLE II

COMPARISONS OF SOLUTION ACCURACY

Function	RCGA			LM_RCGA			RCGA – LM_RCGA
	Mean	Best	Std	Mean	Best	Std	*t*-test
f_1	0.0594	0.0128	0.0259	$\mathbf{8.52\times10^{-255}}$	$\mathbf{2.58\times10^{-264}}$	**0**	22.94968^+
f_2	0.0504	0.016	0.0182	$\mathbf{1.18\times10^{-161}}$	$\mathbf{1.09\times10^{-166}}$	$\mathbf{3.56\times10^{-161}}$	27.67575^+
f_3	**176.323**	**68.186**	**70.036**	3533.48	108.228	2012.15	-16.6831^+
f_4	3.173	1.52	0.929	$\mathbf{5.44\times10^{-22}}$	$\mathbf{4.21\times10^{-25}}$	$\mathbf{1.15\times10^{-21}}$	34.17042^+
f_5	93.064	10.492	**31.744**	**71.420**	**0.679**	37.757	4.38742^+
f_6	**0**	**0**	**0**	**0**	**0**	**0**	0
f_7	8.80×10^{-3}	2.99×10^{-3}	2.87×10^{-3}	$\mathbf{2.94\times10^{-4}}$	$\mathbf{4.45\times10^{-5}}$	$\mathbf{1.20\times10^{-4}}$	29.58898^+
f_8	**-12569.3**	**-12569.4**	**0.0535**	-12562.1	-12568.2	16.854	-4.27645^+
f_9	0.0246	3.33×10^{-3}	9.77×10^{-3}	**0**	**0**	**0**	25.19648^+
f_{10}	0.0608	0.0288	0.0152	$\mathbf{4.89\times10^{-15}}$	$\mathbf{4.14\times10^{-15}}$	$\mathbf{1.45\times10^{-15}}$	40.07716^+
f_{11}	0.121	0.0329	0.0478	$\mathbf{9.51\times10^{-3}}$	**0**	**0.0112**	22.72838^+
f_{12}	3.04×10^{-4}	1.05×10^{-5}	4.31×10^{-4}	$\mathbf{1.54\times10^{-5}}$	$\mathbf{4.89\times10^{-7}}$	$\mathbf{2.54\times10^{-5}}$	6.68504^+
f_{13}	$\mathbf{4.08\times10^{-3}}$	4.80×10^{-4}	$\mathbf{5.20}\times10^{-3}$	6.37×10^{-3}	$\mathbf{1.37\times10^{-5}}$	0.014	-1.53394
f_{14}	-5.562	**-10.1532**	**3.240**	**-6.109**	**-10.1532**	3.258	1.18923
f_{15}	-5.812	-10.4028	3.331	**-5.875**	**-10.4029**	**3.229**	0.13563
f_{16}	**-6.383**	-10.5363	3.491	-5.832	**-10.5364**	**3.392**	-1.13294

$^+$The value of *t* with 198 degrees of freedom is significant at a=0.05 by a two-tailed test.

100 trials, and the horizontal axis is the number of generations. Due to the better global search ability of the random mutation operator, the classical RCGA displays a faster convergence speed than LM_RCGA at the beginning of the training. However, after several generations, the convergence speed of RCGA reduces. This is because that the classical mutation is entirely random and directionless, it could interrupt the convergence of the population when the search direction is determined. On the other hand, LM_RCGA maintains a fast convergence speed all along during the optimization process, and achieves better function values than RCGA in optimizing all the six functions.

Moreover, for each function, an absolute error value is defined and shown in Table III. If the solution value lies within the "Error", the trial is regarded as successful. In this way, the convergence speeds of RCGA and LM_RCGA are compared. As listed in Table III, the column "ErrGen" shows the average generation in which the successful function value within the predefined error value is firstly obtained in the 100 trials. The column "%ok" shows the numbers of successful trials out of the 100 trials. It can be seen that for 11 out of the 16 benchmark functions, LM_RCGA obtains successful results within the error values in a much earlier generation than that of classical RCGA. But due to the better global search ability at the beginning of the training, which we have mentioned before, in dealing with the low-dimensional function f_{14}, f_{15} and f_{16}, classical RCGA obtains successful results faster than LM_RCGA. Moreover, except for f_4 the successful rates in RCGA are all 100%, and except for f_3 the successful rates in LM_RCGA are all 100%.

Fig. 5 shows the histograms of the numbers of successful trials in solving the f_1, f_6, f_7, and f_9. The vertical axis is the statistical number of success trials out of the 100 trials, and the horizontal axis is the number of generations. It can be observed that the convergence speed of the LM_RCGA is faster than that of RCGA.

TABLE III

COMPARISONS OF CONVERGENCE RATE

Function	Error	RCGA		LM_RCGA	
		ErrGen	%ok	ErrGen	%ok
f_1	1	2345	100	**183**	100
f_2	1	874	100	**129**	100
f_3	10000	**163**	**100**	1728	99
f_4	5	2339	97	**501**	100
f_5	2000	333	100	**156**	**100**
f_6	200	214	100	**94**	100
f_7	0.2	138	100	**136**	100
f_8	-12000	**148**	100	859	100
f_9	5	664	100	**392**	100
f_{10}	1	1170	100	**178**	100
f_{11}	1	1814	100	**173**	100
f_{12}	1	284	100	**149**	100
f_{13}	5	315	100	**140**	100
f_{14}	-2	**8**	100	38	100
f_{15}	-1	**2**	100	7	100
f_{16}	-1	**1**	100	5	100

V. CONCLUSION

In this paper, a novel mutation operator based on a linear map for RCGAs is proposed. The mutation operator is performed by using a randomly generated mapping matrix. The mapping matrix should be an orthogonal matrix so that the mutation preserves the information of the original chromosome and treats every dimension without bias. We introduce a rotation-like matrix and applied the matrix in the experiments. 16 benchmark functions are tested. Results and comparisons demonstrate that RCGA with the linear map-based mutation scheme (LM_RCGA) outperforms the classical RCGA.

It should be emphasized that the constraints and mapping matrix used in the linear map-based mutation can be defined in various ways. The proposed orthogonal rotation-like matrix is only one of the possible ones. Future work is that we can test some other mapping matrixes to utilize the interaction between genes of different dimensions on the chromosome in order to further improve LM_RCGA.

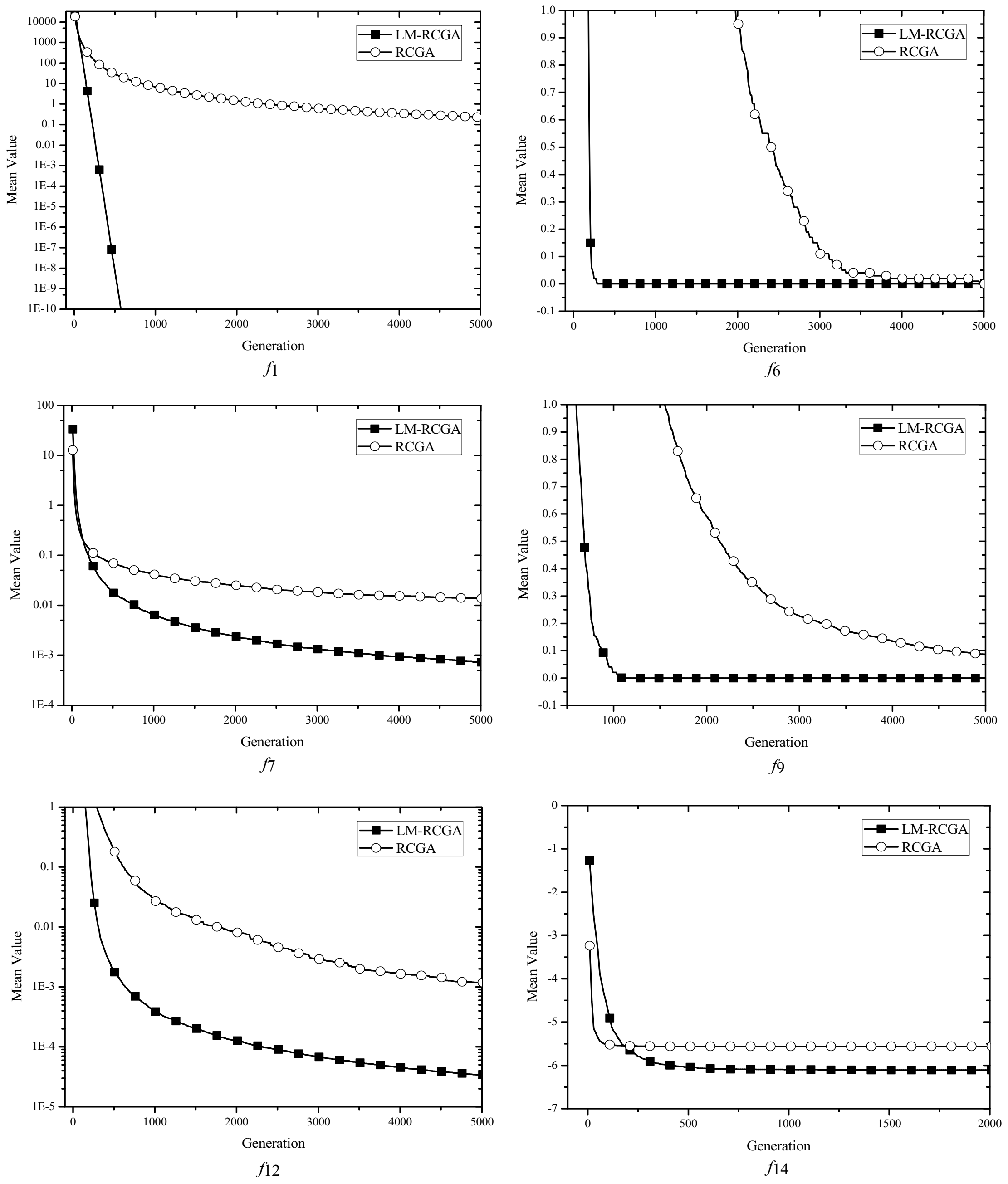

Fig. 4. The convergence curves of the mean values during the training process.

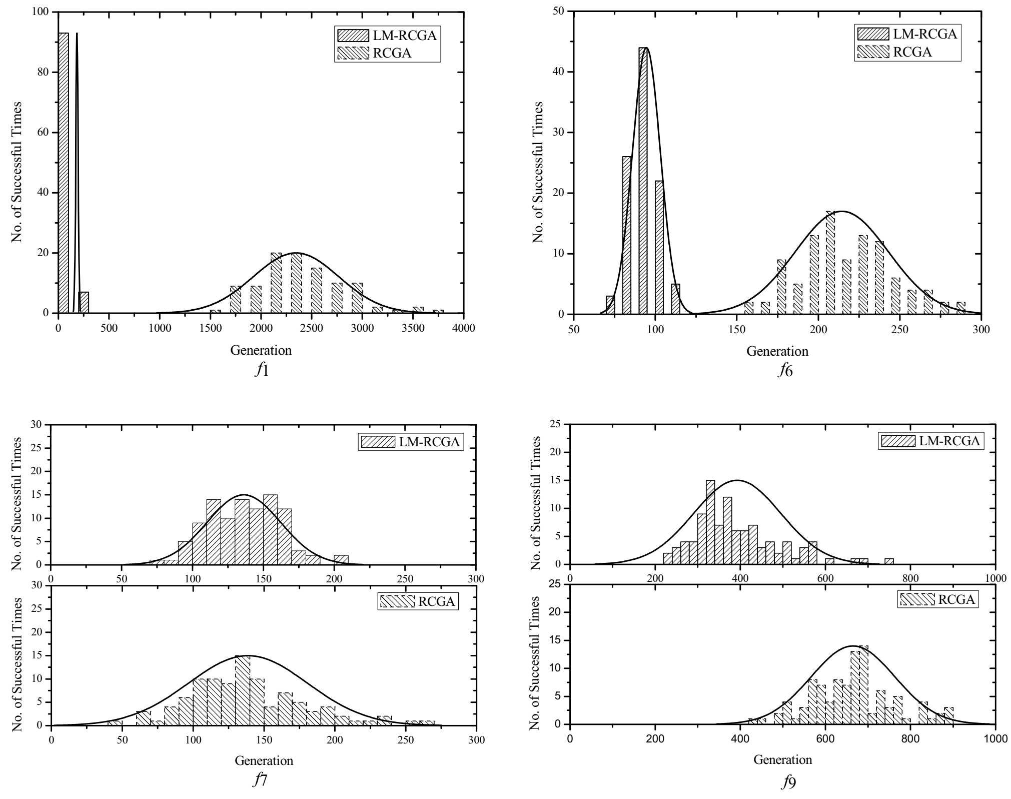

Fig. 5. The histograms of the numbers of successful trials.

References

[1] J. H. Holland, *Adaptation in Neutral and Artificial Systems*, the University of Michigan, 1975.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York, 1989.

[3] F. Herrera, M. Lozano and J. L. Verdegay, "Tackling real-coded genetic algorithms: operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265-319, 1998.

[4] D. E. Goldberg, "Real-coded genetic algorithms, virtual alphabets, and blocking," *Complex Systems*, vol. 5, pp. 139–167, 1991.

[5] Y. Lin and J. Zhang, "A contour method in population-based stochastic algorithms," *Proceeding of IEEE Congress on Evolutionary Computation 2008 (CEC'08)*, pp. 2393-2400, 2008.

[6] J. Zhang, J.-H. Zhong, and X.-M. Hu, "A novel genetic algorithm with orthogonal prediction for global numerical optimization," *X. Li et al. (Eds.): SEAL 2008, LNCS 5361*, pp. 31-40, 2008.

[7] J. Zhang, H. Chung and W. L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms", *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 326-335, 2007.

[8] L. Du, J. Bigham and L. Cuthbert, "Towards intelligent geographic load balancing for mobile cellular networks," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 33, no. 4, pp.480-491, 2003.

[9] N. Amjady and H. Nasiri-Rad, "Nonconvex economic dispatch with AC constraints by a new real coded genetic algorithm," *IEEE Transactions on Power Systems*, vol. 24, no. 3, pp. 1489 – 1502, 2009.

[10] S. Grubisic, W. P. Carpes and J. P. A. Bastos, "Optimization model for antenna positioning in indoor environments using 2-D ray-tracing technique associated to a real-coded genetic algorithm," *IEEE Transactions on Magnetics*, vol. 45, no. 3, pp. 1626-1629, 2009.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York, 1992.

[12] F. Li and T. Zhang, "Study on genetic algorithm based on schema mutation and its performance analysis," *2009 Second International Symposium on Electronic Commerce and Security*, vol. 1, pp. 548-551, 2009.

[13] M.-Y. Cui, "An improved float-coded genetic algorithm based on wavelet denoising mutation," *7th World Congress on Intelligent Control and Automation, WCICA 2008*, pp. 4018 – 4023, 2008.

[14] G. W. Greenwood, "Adapting mutations in genetic algorithms using gene flow principles," *The 2003 Congress on Evolutionary Computation, CEC '03*, vol. 2, pp. 1392 – 1397, 2003.

[15] J.-X. Peng, K. Li and S. Thompson, "A combined adaptive bounding and adaptive mutation technique for genetic algorithms," *Fifth World Congress on Intelligent Control and Automation, WCICA 2004*, vol. 3, pp. 2154 – 2158, 2004.

[16] J. Teo, "Self-adaptive mutation for enhancing evolutionary search in real-coded genetic algorithms," *International Conference on Computing & Informatics, ICOCI '06*, pp. 1-6, 2006.

[17] S.-Y. Yuen and C.-K. Chow, "A genetic algorithm that adaptively mutates and never revisits," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 454-472, 2009.

[18] H. Mühlenbein and D. Schlierkamp, "Predictive models for the breeder genetic algorithm I. continuous parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 25-49, 1993.

[19] H. M. Voigt and T. Anheyer, "Modal mutations in evolutionary algorithms," *The 1st IEEE Conference on Evolutionary Computation. Part 1 (of 2)*, pp. 88-92, 1994.

[20] P. R. Halmos, *Finite-Dimensional Vector Spaces*, Springer-Verlag, New York, 1994.

[21] H. E. Rose, *Linear Algebra: A Pure Mathematical Approach*, Birkhäuser Basel, 2002.

[22] X. Yao, Y. Liu, and G.-M. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82-102, 1999.