International Journal of Innovative
Computing, Information and Control
Volume 11, Number 5, October 2015

 ISSN 1349-4198
pp. 1811–1821

THE MULTI-CHILD GENETIC ALGORITHM

Fulin Wang*, Xiaoming Fu and Huixia Zhu

College of Engineering
Northeast Agricultural University
No. 59, Mucai Street, Xiangfang District, Harbin 150030, P. R. China
*Corresponding author: fulinwang1462@126.com

Received April 2015; revised August 2015

Abstract. *In this paper, the multi-child genetic algorithm (MCGA) is first proposed to enhance the solution searching abilities and increase the computational speed of genetic algorithms. The advantage of MCGA is verified based on schema theorem in this article. The MCGA adopts a method for producing multi-child offspring to preserve chromosomes which are excellent. The method also improves the chance of generating superior chromosomes to assist the solution exploration. The MCGA also uses the operating method of population evolution. This method solves the problem that in traditional genetic algorithm excellent chromosomes in offspring produced by crossover may be damaged and unable to survive in the process of mutation. Experiments were conducted on typical problems which include non-linear, uni-modal, multi-modal and pathological functions. The results showed that the average computational time for MCGA is 1.8 to 5.1 times faster than the traditional GA and the average number of iteration steps for MCGA is 24% to 45% of the traditional GA, when the population entirely converges toward global optimal solutions under the given condition. It is proved that MCGA has stronger global searching capacity and higher convergence rate compared to traditional genetic algorithm.*
Keywords: Multi-child genetic algorithm, Biological evolution, Schema theorem, Population evolution

1. **Introduction.** A genetic algorithm (GA), which is based on natural selection and evolutionary principle, is a highly parallel, stochastic, self-adaptive and heuristic search technique used in computing to find true or approximate solutions to optimization and search problems [1-3]. John Holland, the professor of the University of Michigan, invented GA and he first promulgated this idea in his book "Adaptation in Natural and Artificial Systems" in the year 1975 [4-6]. Afterwards, many scholars conducted thorough research on GA and pushed forward the development of evolutionary algorithms. So far various kinds of improved genetic algorithms have been proposed, such as hierarchical GA, CHC algorithms, messy GA, adaptive GA, multi-niche crowding GA, hybrid GA and parallel GA [7-12]. In these researches, two or multiple parents generate double individuals (chromosomes) [13-15], or one parent creates an individual [16-19]. Namely, the number of new individuals produced by parents is equal to or less than the number of the parents. However, in the process of biological evolution, the number of children is normally more than the number of their parents, which is beneficial to not only the survival of the species but increasing the probability of generating better individuals for the species. Thus, this species has a higher ability to adapt to environment.

In GA, looking for the best solution depends on new born chromosomes, which are generated by crossover and mutation operations to have higher ability (fitness) of finding better solutions. When the solution searching comes to a standstill, superior (potential) chromosomes are required in order to break free from the local extremum, explore

unsearched areas and speed up the solution searching progress. If most of the new born offspring, which have low fitness, are eliminated by selection, the solution searching progress may slow down and even halt at a standstill. It is an important issue for parents to generate better individuals in offspring as a means of speeding up the solution searching process. Since an increase in the number of new born chromosomes enhances the probability of generating better individuals, the quantity of new born chromosomes will affect GA's solution exploration ability directly. In order to efficiently drive the population and improve the rate of convergence toward optimum solution for GA, the multi-child genetic algorithm (MCGA) is proposed in this paper based on this theory of biological evolution. The MCGA is including generating multi-child offspring and population evolution, for enhancing the solution searching abilities and increasing the computational speed of genetic algorithms.

The rest of this paper is organized as follows. The concept of MCGA is presented in Section 2. The MCGA is analyzed based on schema theorem in Section 3. A method of producing multi-child offspring for MCGA is given in Section 4. The population evolution of MCGA is described in Section 5. The test functions, experimental settings, test results and analysis are presented in Section 6. Finally, some conclusions are drawn in Section 7.

2. Concept of MCGA.

Multi-child genetic algorithm (MCGA) is a new concept, and the definition of MCGA is provided in this article as follows.

Definition 2.1. *The number of new individuals created by parents is more than their parents in genetic algorithm, called as multi-child genetic algorithm (MCGA).*

In general, the number of new individuals is an integral multiple of the number of their parents in MCGA. Representing the number of new individuals by N_1 and the number of their parents by N_0, we have

$$N_1 = \lambda \cdot N_0, \quad \lambda \in \{x | x \geq 2, x \in \mathbb{N}\} \tag{1}$$

where the λ is the proportional coefficient between the number of new individuals and their parents.

Since the number of children is more than their parents', the chance of producing excellent individuals that have high fitness in offspring increases. In competition, a large number of individuals that have strong adaptability survive and a large number of individuals that show less adaptability will be eliminated, which is based on Darwin's principle "Survival of the fittest" [20]. Compared with the traditional GA, there is the fiercer competition among individuals and the higher efficiency of evolution in MCGA. Thus, the convergence rate of the MCGA to optimum solutions is faster than the traditional GA.

3. Analysis of MCGA Based on Schema Theorem.

In order to illustrate the advantages of MCGA, it is necessary to analyze the mathematical mechanism of MCGA based on Schema Theorem, because there are a lot of stochastic operations in MCGA. Some definitions are given as follows before the analysis.

Definition 3.1. *A schema is a template that identifies a subset of strings (chromosomes) with similarities at certain string positions.*

In strings with binary code, a schema is a character string based on a character set which includes three characters $(0, 1, *)$ in which the symbol $*$ represents arbitrary character (0 or 1). For example, the model $\{* \ 1 \ *\}$ describes a subset with four elements $\{010, 011, 110, 111\}$.

Definition 3.2. *The number of certain positions in schema H is called as schema order, denoted by $O(H)$. For example, $O(01**1**) = 3$.*

Definition 3.3. *The distance between the first and the last certain position in schema H is called as defining length, denoted by $\delta(H)$. For example, $\delta(01**1**) = 4$.*

Let us suppose that we have a population of chromosomes (strings) whose total number is represented by n. The population at time t is represented by $A(t)$. The number of chromosomes which include certain schema H in $A(t)$ at time t is represented by $m(H,t)$. In selection process, each chromosome is selected according to its fitness. Representing an arbitrary chromosome in $A(t)$ by A_i, the probability that the string A_i is selected is

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}, \tag{2}$$

where f_i represents the fitness of A_i.

We can thus write the number of strings that include schema H at time $(t+1)$ as

$$m(H,t+1) = m(H,t) \cdot n \cdot \frac{f(H)}{\sum_{j=1}^{n} f_j}, \tag{3}$$

where $f(H)$ represents the average fitness of strings containing schema H at time t.

The average fitness of the population can be written as

$$\bar{f} = \frac{\sum_{j=1}^{n} f_j}{n}. \tag{4}$$

Hence, Equation (3) can be represented as

$$m(H,t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}}. \tag{5}$$

Assuming that beginning with $t = 0$ the fitness of chromosome containing schema H is invariably higher than the average fitness of the population, putting $f(H) - \bar{f} = c\bar{f}$, where c is a constant, the equation of schema growth can be written as

$$m(H,t+1) = m(H,t) \cdot \frac{\left(\bar{f} + c\bar{f}\right)}{\bar{f}} = (1+c) \cdot m(H,t) = (1+c)^t \cdot m(H,0). \tag{6}$$

From (6), it is shown that in the operation of selection schema of which fitness is higher than the average fitness of population shows up as exponential growth. As with traditional GAs, obviously MCGA does not produce new schemata but increase the number of existing schemata with high fitness. Therefore, we need to take crossover (recombination) operation.

In order to illustrate the effect of crossover on schema in MCGA, we first take the traditional GA with single point crossover for example. Let us suppose that we have a specified string C whose length is seven. And two representative schemata H_1 and H_2 are contained in string C, as follows.

$$\begin{aligned} C &= 1\,0\,0\,0\,1\,1\,1 \\ H_1 &= *\,0\,*\,*\,*\,*\,1 \\ H_2 &= *\,*\,*\,0\,1\,*\,* \end{aligned} \tag{7}$$

Assuming that string C is selected to participate in crossover operation and a crossover point is selected randomly, if the crossover point is between the third and fourth position

of this string, we analyze the effect of crossover on schemata H_1 and H_2. Representing cross-site by '|', (7) can be written as

$$\begin{aligned} C &= 1\,0\,0\,|\,0\,1\,1\,1 \\ H_1 &= *\,0\,*\,|\,*\,*\,*\,1 \\ H_2 &= *\,*\,*\,|\,0\,1\,*\,* \end{aligned} \tag{8}$$

Obviously, schema H_1 will be destroyed, if the certain position of schema H_1 in the object that is crossed over with string C is different with string C. However, with the same cross-site, schema H_2 will be kept in a string in offspring. Thus, the schema H_1 is more difficult to survive than H_2. Because the cross-site is randomly selected, the probabilities that each cross-site is randomly selected are equal. Since $\delta(H_1) = 5$ and $\delta(H_2) = 1$ according to Definition 3.2, the probabilities that schemata H_1 and H_2 are destroyed are respectively 5/6 and 1/6. In general, the probability that schema H is destroyed in traditional GA can be written as

$$p_{d(traditionalGA)} = \frac{\delta(H)}{l-1}, \tag{9}$$

where $\delta(H) \leq l - 1$.

From (1), in MCGA, the number of children is λ times than their parents'. Thus, a pair of parents needs to produce children whose number is 2λ. In order to create 2λ individuals, the both parents must be crossed at stochastic cross-sites λ times. If we suppose that one of the both parent's chromosomes contains schema H, the chance that schema H is destroyed in MCGA can be represented as

$$p_{d(MCGA)} = \left(\frac{\delta(H)}{l-1}\right)^{\lambda}, \tag{10}$$

where $\delta(H) \leq l - 1$.

It is worth remarking from (10) that with the increase of the value of λ, the probability that schema H is destroyed is constantly reduced. From (9) and (10), we evidently have

$$\left(\frac{\delta(H)}{l-1}\right)^{\lambda} < \frac{\delta(H)}{l-1}, \tag{11}$$

where $\lambda \in \{x | x \geq 2, x \in \mathbb{N}\}$.

Namely,

$$p_{d(MCGA)} < p_{d(traditionalGA)}. \tag{12}$$

Because crossover is stochastic process, representing the crossover probability by p_c, we can write the lower boundary of survival probability of schema H in MCGA as

$$p_{LS(MCGA)} = 1 - p_c \cdot \left(\frac{\delta(H)}{l-1}\right)^{\lambda}. \tag{13}$$

Considering the effect of the selection and crossover on schema in MCGA, because selection and crossover are not related, the lower boundary of the number of schema H in offspring can be obtained as

$$m_{L(MCGA)}(H, t+1) = m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \left(\frac{\delta(H)}{l-1}\right)^{\lambda}\right]. \tag{14}$$

From (11) and (14), we evidently have

$$m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \left(\frac{\delta(H)}{l-1}\right)^{\lambda}\right] > m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1}\right], \tag{15}$$

where $\lambda \in \{x | x \geq 2, x \in \mathbb{N}\}$, $\delta(H) \leq l-1$.

Obviously, in (15), $m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1}\right]$ is the lower boundary of the number of schema H in traditional GA at time $(t+1)$.

Putting $m(H,t) \cdot \frac{f(H)}{\bar{f}} \cdot \left[1 - p_c \cdot \frac{\delta(H)}{l-1}\right] = m_{L(traditionalGA)}(H, t+1)$, we have

$$m_{L(MCGA)}(H, t+1) > m_{L(traditionalGA)}(H, t+1). \tag{16}$$

From (11) and (12), supposing that the schema H just has excellent chromosomes, the chance that the schema H is destroyed in MCGA is significantly less than in the traditional GA. Hence, the MCGA can efficiently preserve chromosomes which are excellent currently.

It is shown from (16) that, when we suppose that the schema H is a superior (potential) schema whose fitness is higher than the average fitness of population, the number of schema H in MCGA is significantly more than in traditional GA at time $(t+1)$, if the numbers of schema H in MCGA and traditional GA are all $m(H,t)$ at time t. As a result, compared with the traditional GA, MCGA has the higher efficiency of population evolution and the faster rate of convergence toward optimum solution. To verify the above conclusions, this study presents a method for producing multi-child offspring of MCGA.

4. **The Method of Producing Multi-Child in MCGA.** Let us suppose that we have $Parent_1$ and $Parent_2$ that represent respectively two different parent chromosomes which are selected from a population according to their fitness to take part in the crossover. The coding lengths of $Parent_1$ and $Parent_2$ are l, and then the number of crossover point selected randomly is n $(n \leq l-1)$, so the chromosomes $Parent_1$ and $Parent_2$ are divided into $n+1$ segments, namely $Parent_1 = A_1A_2A_3 \cdots A_k \cdots A_{n-1}A_nA_{n+1}$, $Parent_2 = B_1B_2B_3 \cdots B_k \cdots B_{n-1}B_nB_{n+1}$. The method of producing multi-child is as follows.

The two segments on both sides of each crossover point of parent chromosomes are regarded as substrings.

(1) $A_2A_3 \cdots A_k \cdots A_{n-1}A_nA_{n+1}$ and $B_2B_3 \cdots B_k \cdots B_{n-1}B_nB_{n+1}$ are two substrings after first crossover point of two parent chromosomes. They are exchanged to produce two individuals $Child_1$ and $Child_2$, so we have obviously

$$Child_1 = A_1B_2B_3 \cdots B_k \cdots B_{n-1}B_nB_{n+1} \text{ and } Child_2 = B_1A_2A_3 \cdots A_k \cdots A_{n-1}A_nA_{n+1}.$$

(2) Double substrings $A_3 \cdots A_k \cdots A_{n-1}A_nA_{n+1}$ and $B_3 \cdots B_k \cdots B_{n-1}B_nB_{n+1}$ after second crossover point of these two parent chromosomes are exchanged to produce two individuals $Child_3$ and $Child_4$, so we have

$$Child_3 = A_1A_2B_3 \cdots B_k \cdots B_{n-1}B_nB_{n+1} \text{ and } Child_4 = B_1B_2A_3 \cdots A_k \cdots A_{n-1}A_nA_{n+1}.$$

(3) In the same way, we finally have $Child_{2n-1} = A_1A_2A_3 \cdots A_k \cdots A_{n-1}A_nB_{n+1}$ and $Child_{2n-1} = B_1B_2B_3 \cdots B_k \cdots B_{n-1}B_nA_{n+1}$. As a result, we can obtain the offspring made up of many children which are produced by crossing over the parents. The general equation of arbitrary pair of individuals in the offspring can be written as

$$\begin{cases} Child_{2k-1} = A_1A_2A_3 \cdots A_kB_{k+1} \cdots B_{n-1}B_nB_{n+1} \\ Child_{2k} = B_1B_2B_3 \cdots B_kA_{k+1} \cdots A_{n-1}A_nA_{n+1} \\ n \leq l-1 \end{cases}, \tag{17}$$

where l is the coding length of parent chromosomes.

From (17), it is shown that the number of new individuals is $2n$ which is n times the number of parents, namely the proportional coefficient between the number of new individuals and their parents is equal to the number of crossover points selected randomly (namely $\lambda = n$). The method of producing multi-child offspring in MCGA is shown in Figure 1.

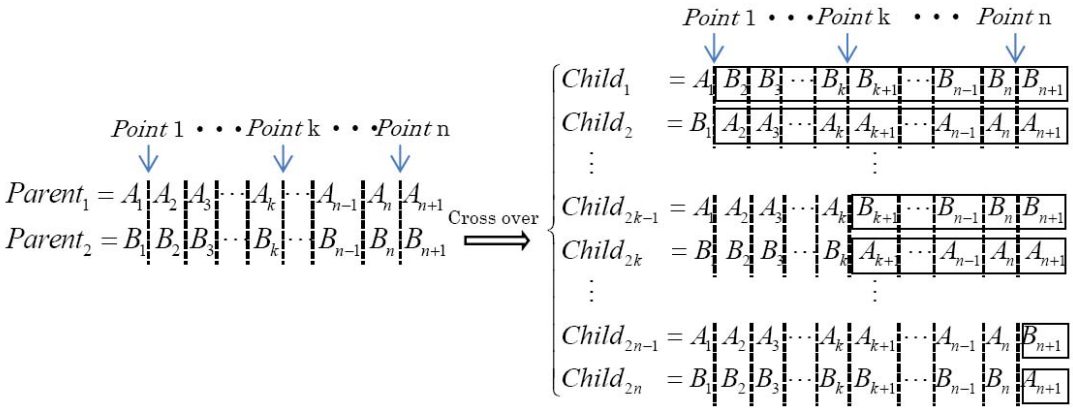


FIGURE 1. The method of producing multi-child offspring in MCGA

In Figure 1, $Child_i$ is the i-th individual in the offspring formed by crossing over the parents. It is demonstrated in Figure 1 that crossing over a pair of parents can create $2n$ new individuals (multi-child offspring) in MCGA. The number of producing new individuals in MCGA is n times more than in traditional GA.

In MCGA, for example, the number of crossover points randomly selected is $n = 2$ (namely $\lambda = 2$), and two different parent chromosomes which are selected according to their fitness are respectively $Parent_1$ and $Parent_2$. These two parent chromosomes are encoded by bit strings as $Parent_1 = 100001000111000$ and $Parent_2 = 111110110011011$. Supposing that two crossover points $Point_1$ and $Point_2$ which are randomly selected are located respectively in between the sixth and seventh bit binary code and between the twelfth and thirteenth bit binary code, we then have

$Point_1$ $Point_2$

$Parent_1$=100001 | 000111 | 000 ,
$Parent_2$=111110 | 110011 | 011 .

Four new individuals created by crossing over $Parent_1$ and $Parent_2$ in the way shown in Figure 1 are

$Point_1$ $Point_2$

$Child_1$=100001 | 110011 | 011;
$Child_2$=111110 | 000111 | 000;
$Child_3$=100001 | 000111 | 011;
$Child_4$=111110 | 110011 | 000.

Thus, the number of producing new individuals will greatly go up in MCGA, which increases the possibility of producing more of the best individuals. Therefore, MCGA has faster convergence speed and greater capability to find better solutions compared with traditional GA. It should be noted that, vast increase in the number of producing new individuals could lead to increasing computational complexity of algorithm program, which has an effect on computational speed of MCGA instead. As a result, it is extremely important to take an advisable value of the proportional coefficient λ. How to select the proportional coefficient λ needs to be further studied and explored. The smallest proportional coefficient $\lambda = 2$ will be taken in this paper, in order to verify the superiority of MCGA by comparing with the traditional GA.

5. **The Population Evolution of MCGA.** The main basis of the population evolution of MCGA is as follows.

(1) The population size is constant, which can be explained from the view of Biology that the number of living individuals must not be beyond some constant size because of the limit of population capacity.

(2) The population should maintain an appropriate degree of diversity.

(3) An excellent individual with high fitness value has a great chance of being selected to generate children and a weak member with low fitness may be eliminated in the selection under the limit of population capacity.

According to the above three points, the operating method of population evolution of MCGA is presented as follows. Firstly, create the initial population with a size of n_0 randomly and set the crossover probability $p_c = 100\%$. Secondly, take the initial population as parent population and calculate the fitness of each individual so as to preserve a few elite individuals with the highest fitness (we suppose that the number of elites is m). Then cross over the rest of parents to form new multi-child offspring, in

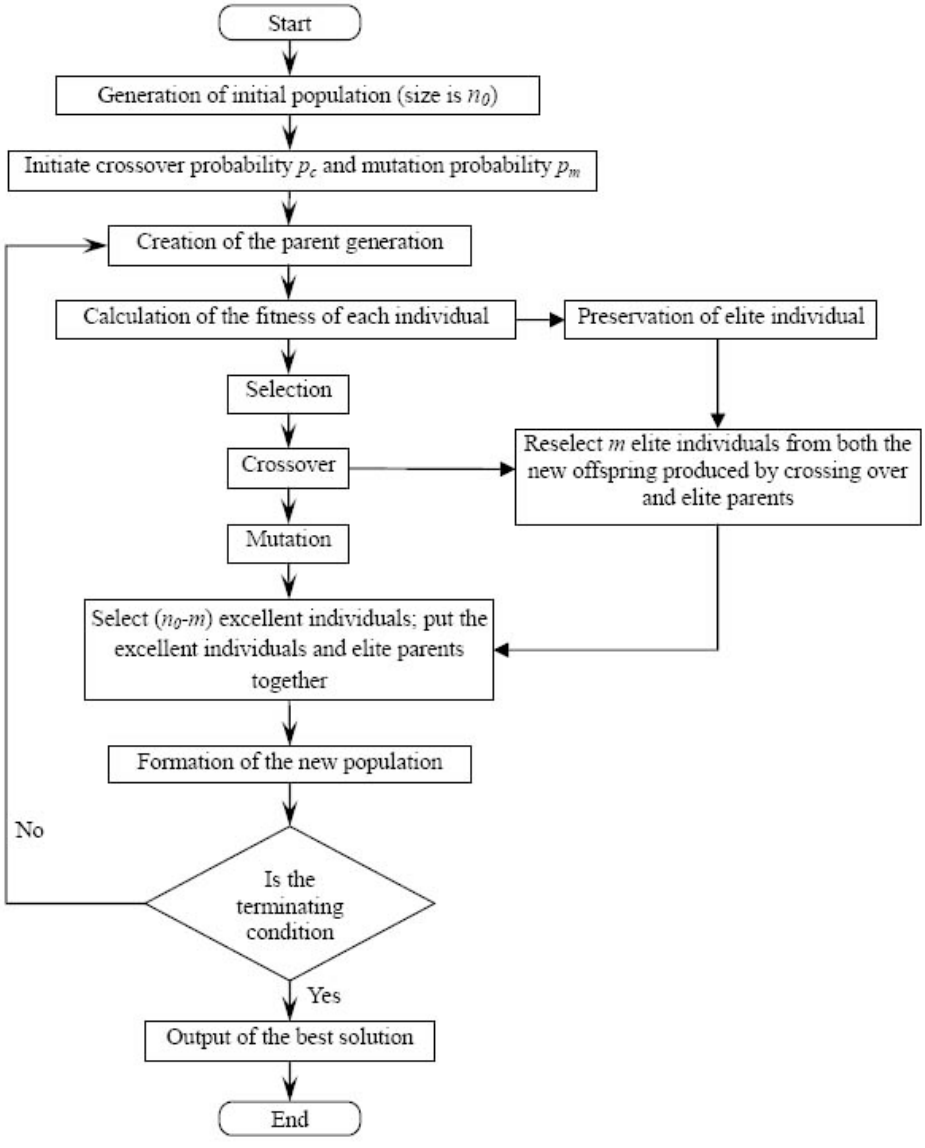


FIGURE 2. Basic flowchart of the population evolution of MCGA

which the number of individuals is n_1 ($n_1 > n_0$). Thirdly, reselect m elite individuals from both the new offspring and elite parents. Fourthly, with a mutation probability, mutate new offspring at each locus (position in chromosome). Fifthly, pick $(n_0 - m)$ excellent individuals from the new offspring according to their fitness. Then put the excellent individuals and elite individuals together to form a new population. Sixthly, if the end condition is satisfied, stop, and return the best solution in current population. The flowchart for the population evolution of MCGA is shown in Figure 2.

In the above population evolution of MCGA, the number of new individuals created by crossing over parents increases obviously because of taking the crossover probability $p_c = 100\%$, which increases the possibility of producing better excellent individuals with higher fitness. Although schemata with high fitness values could be damaged in the crossover operation, the excellent chromosomes in the parents are copied into the new population due to the preservation of elite individuals. Thus, taking the crossover probability of 100% can improve the computational speed of GA. In addition, this population evolution of MCGA solves the problem that in traditional genetic algorithm excellent individuals in the offspring made by crossover may be damaged and unable to survive in the process of mutation. Meanwhile, there are $(n_0 - m)$ excellent individuals in the new population after mutation, which help to maintain diversity in the population.

6. Testing and Analyzing of MCGA.

6.1. The selection of the test functions and parameters. Four commonly used complicated test functions are selected to compare the performance of MCGA proposed in this article with the traditional GA. These typical test functions which have multiple local extreme points include varieties of mathematical characteristics such as continuous, non-linear, uni-modal, multi-modal, and pathological.

Test function 1:

$$\min f_1(x, y) = 100(y - x^2)^2 + (x - 1)^2, \quad -10 \le x, y \le 10. \tag{18}$$

This function, which is called Rosenbrock function or Rosenbrock's valley, is non-convex function used as a performance test problem for optimization algorithms [21]. The global minimum is inside a long, narrow, parabolic shaped flat valley. The 3-D plot of the function is shown in Figure 3(a). To find the valley is trivial. To converge to the global minimum, however, is difficult. It has a global minimum at $(x, y) = (1, 1)$, where $f(x, y) = 0$.

Test function 2:

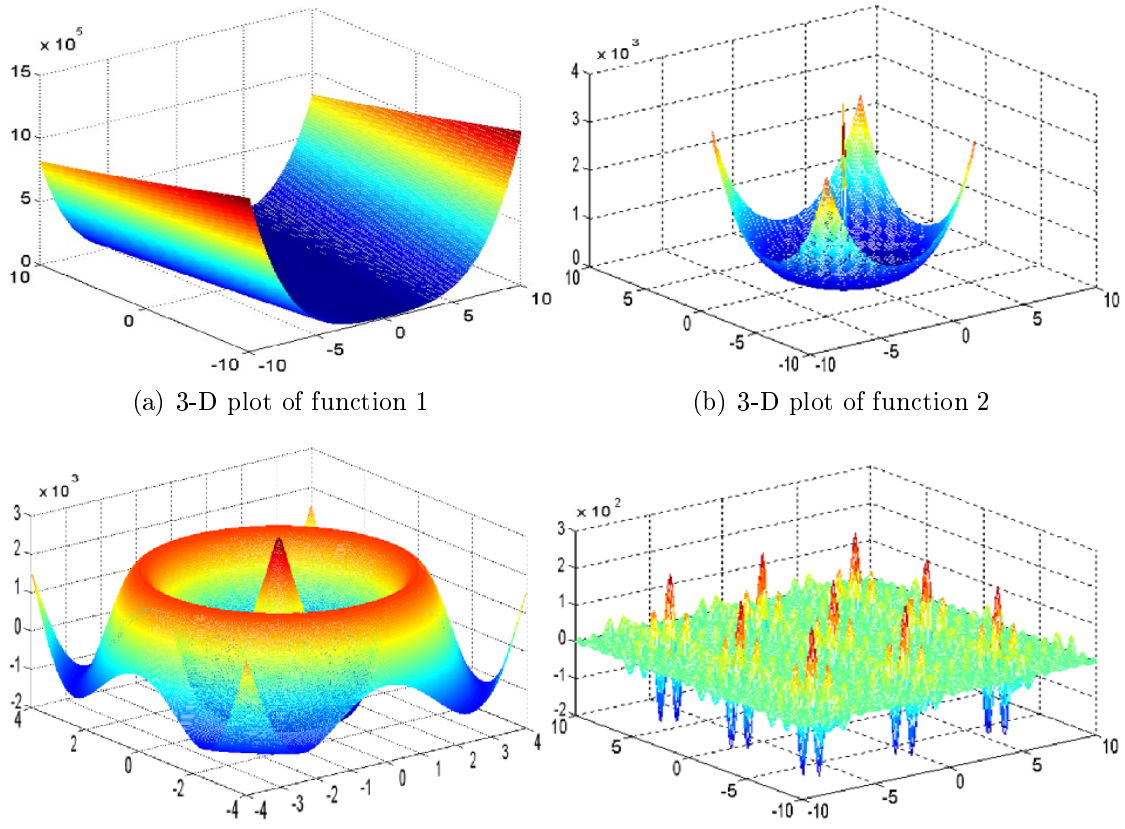
$$\max f_2(x, y) = \left[\frac{3}{0.05 + (x^2 + y^2)}\right]^2 + (x^2 + y^2)^2, \quad -5.12 \le x, y \le 5.12. \tag{19}$$

This function has only one global maximum point surrounded by multiple minima. Besides, the function has four local maximum points which makes it very difficult to find the global maximum point of this function. To search the global maximum point is like to find a needle in a haystack, which is extremely difficult. The global maximum value is $f(0, 0) = 3600$. The 3-D plot of function 2 is shown in Figure 3(b).

Test function 3:

$$\max f_3(x, y) = \frac{3600\left(\sin^2\sqrt{x^2 + y^2} - 0.5\right)}{\left[1 + 0.001\left(x^2 + y^2\right)\right]^2} + \left[\frac{3}{0.1 + (x^2 + y^2)}\right]^2 + \left(x^2 + y^2\right)^2, \tag{20}$$
$$-4 \le x, y \le 4.$$

This function has unique global maximum point of which value is 2700. However, the global maximum point is surrounded by infinite number of local maximum points whose



(a) 3-D plot of function 1

(b) 3-D plot of function 2

(c) 3-D plot of function 3

(d) 3-D plot of function 4

FIGURE 3. Schematic diagram of each test function

values are all 2698.6. These points constitute a ring shape of peaks which looks like a crater. Besides, the function also has four local maximum points, which is shown in Figure 3(c). As a result, the searching of global maximum point of the function is liable to be trapped in the local maximum points.

Test function 4:

$$\min f_4(x,y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x+i]\right\}\left\{\sum_{i=1}^{5} i\cos[(i+1)y+i]\right\} + 0.5\left[(x+1.42513)^2+(y+0.80032)^2\right], \quad -10 \le x,y \le 10. \tag{21}$$

This function has seven hundred and sixty local minima of which only one global minimum is $f(-1.42513,-0.80032) = -186.7309$. The 3-D plot is shown in Figure 3(d).

In order to ensure the validity of testing, in the testing procedure, both MCGA and traditional GA adopt the binary coding strategy. The initial population of size 100 is selected at random, but any number of populations can be elected based on the requirement and application. The precision of coding of every decision variable is set to be 10^{-6}; the crossover rate is $p_c = 100\%$; the mutation rate is $p_m = 10\%$; the number of elite individuals is $m = 10$. In MCGA, the value of parameter λ is set to be 2. The terminating conditions of these two GAs are all

$$|f_i^z - f_i^*| \le \varepsilon_i, \quad (i = 1,2,3,4), \tag{22}$$

where f_i^* is used to represent the theoretical global optimal solution of the test function i; f_i^z is used to represent the global optimal solution of the test function i searched by algorithms; ε_i is used to represent the given searching accuracy. The ranges of decision

variables in four test functions have been given in Equations (18)-(21). The searching accuracies of these functions are all 10^{-4}. The testing of searching algorithm for each function with the two GAs is carried out in the same computer for 1000 times.

6.2. **Testing result and analysis.** The testing result of the four test functions is shown in Table 1. From Table 1, it can be seen that not only the average computational time but also the average number of iteration steps of the MCGA is obviously superior to the traditional GA.

TABLE 1. Testing result of multi-child genetic algorithm (MCGA) and traditional genetic algorithm (GA)

Function	Searching Algorithm	Average Computational Time/s	Longest Computational Time/s	Shortest Computational Time/s	Average Number of Iteration Steps	Maximum Number of Iteration Steps	Minimum Number of Iteration Steps
f_1	Traditional GA	2.9527	7.9301	0.7767	349.42	656	85
	MCGA	1.1109	4.3764	0.1556	150.93	464	16
f_2	Traditional GA	1.8043	4.6677	0.7423	230.79	531	123
	MCGA	0.7804	2.0489	0.1645	104.68	324	21
f_3	Traditional GA	1.0592	3.9836	0.4208	139.77	502	84
	MCGA	0.5824	1.1223	0.1109	63.31	146	11
f_4	Traditional GA	9.7398	37.3043	1.5632	943.80	4376	102
	MCGA	1.8966	10.8650	0.1221	232.20	2310	17

Notes: 1. The testing environment: Intel Core i5 M340 @2.27GHz, 2GB RAM, Microsoft Win7 OS;
2. Compiling environment: MatLab7.11.0 (R2010b).

The average computational time for finding the minimum point of test function f_1 is 1.1109s for MCGA which is 2.66 times faster than 2.9527s for the traditional GA. For the MCGA, the average number of iteration steps for searching convergence of test function f_1 is 150.93, which is 43.2% of the number 349.42 for the traditional GA. For test function f_2, the average computational time for MCGA is 0.7804s which is 2.3 times faster than 1.8043s for the traditional GA. For the MCGA, the average number of iteration steps for searching convergence of test function f_2 is 104.68, which is 45.4% of the number 230.79 for the traditional GA. The average computational time for test function f_3 is 0.5824s and 1.0592s; the MCGA has a 1.82 times faster computational speed. The average number of iteration steps is respectively 63.31 and 139.77 for the two GAs; the MCGA has 2.21 times less iteration steps. The average computational time for test function f_4 is 1.8966s and 9.7398s; the MCGA has a 5.14 times faster computational speed. The average number of iteration steps is respectively 232.2 and 943.8 for the two GAs; the MCGA has 4.06 times less iteration steps. As a result, it can be concluded that the MCGA has much faster computational speed and higher convergence rate than the traditional GA. Consequently, it is shown that the MCGA is better than the traditional GA.

7. **Conclusions.** The MCGA has been presented in this paper based on the theory of biological evolution to enhance the solution searching abilities and increase the computational speed of genetic algorithms. Through analysis of MCGA based on schema theorem, the number of new born chromosomes in MCGA is obviously more than in traditional GA, which not only preserves current chromosomes which are excellent but improves the chance of generating superior chromosomes to assist the solution exploration. The proposed method for producing multi-child offspring of MCGA can effectively increase the number of new born chromosomes. The proposed operating method of population evolution solved the problem that in traditional genetic algorithm excellent individuals in the offspring made in crossover operation may be damaged and unable to survive in the process of mutation. Four typical test functions were experimented. The experiments

show that the average computational time for MCGA is 1.8 to 5.1 times faster than the traditional GA and the average number of iteration steps for MCGA is 24% to 45% of the traditional GA, when the population entirely converges toward global optimal solutions under the given condition. Thus, the proposed MCGA is more efficient to close to optimal solutions than the traditional GA.

Acknowledgment. This research was supported by the National Natural Science Foundation of China (31071331) and National Social Science Foundation of China (13BJY098).

REFERENCES

[1] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*, Springer-Verlag Berlin Heidelberg, New York, 2008.

[2] D. Liu and S. Xu, Inward-outward crossover based genetic algorithm for constrained optimization problem, *Systems Engineering-Theory & Practice*, vol.32, no.1, pp.811-822, 2014.

[3] S. Velkatraman and G. G. Yen, A genetic framework for constrained optimization using genetic algorithms, *IEEE Transactions on Evolutionary Computation*, vol.9, no.4, pp.424-435, 2005.

[4] N. Lin and H. Liu, Dynamic route guidance algorithm based on improved Hopfield neural network and genetic algorithm, *International Journal of Innovative Computing, Information and Control*, vol.10, no.2, pp.811-822, 2014.

[5] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[6] T. Dede, S. Bekiroglu and Y. Ayvaz, Weight minimization of trusses with genetic algorithm, *Applied Soft Computing*, vol.11, no.2, pp.2565-2575, 2011.

[7] S. M. R. Loghmanian, R. Yusof, M. Khalid and F. S. Ismail, Polynomial NARX model structure optimization using multi-objective genetic algorithm, *International Journal of Innovative Computing, Information and Control*, vol.8, no.10(B), pp.7341-7362, 2012.

[8] L. Yao and C.-C. Lin, On a genetic algorithm based scheduled fuzzy PID controller, *International Journal of Innovative Computing, Information and Control*, vol.5, no.10(B), pp.3593-3602, 2009.

[9] F. Wang, J. Wang, C. Wu and Q. Wu, The improved research on actual number genetic algorithms, *Journal of Biomathematics*, vol.21, no.1, pp.153-158, 2006.

[10] L. Wang, H. Wu, F. Tang, D. Zheng and Y. Jin, Hybrid quantum genetic algorithms and performance analysis, *Control and Decision*, vol.20, no.2, pp.156-158, 2005.

[11] C.-Y. Liu, An improved adaptive genetic algorithm for the multi-depot vehicle routing problem with time window, *Journal of Networks*, vol.8, no.5, pp.1035-1042, 2013.

[12] T. M. Cheng and R. Z. Yan, Integrating messy genetic algorithms and simulation to optimize resource utilization, *Computer-Aided Civil and Infrastructure Engineering*, vol.24, no.6, pp.401-415, 2009.

[13] A. E. Eiben, Multi-parent recombination in evolutionary computing, *Advances in Evolutionary Computing, Natural Computing Series*, pp.175-192, 2003.

[14] P. Li, J. Wu, J. Zheng and N. Hu, Theoretical analysis and application research on multi-parent genetic algorithm, *Computer Engineering and Design*, vol.27, no.4, pp.581-583, 2006.

[15] I. Saracoglu, S. Topaloglu and T. Keskinturk, A genetic algorithm approach for multi-product multi-period continuous review inventory models, *Expert Systems with Applications*, vol.41, no.18, pp.8189-8202, 2014.

[16] M. Li, *Theory and Application Research on Partheno-Genetic Algorithm*, Hunan University, Changsha, 2002.

[17] J. Wu and H. Wang, Partheno genetic algorithm for the founder sequence reconstruction problem, *Journal of Computers (Finland)*, vol.8, no.11, pp.2934-2941, 2013.

[18] K. Katayama and H. Narihisa, On fundamental design of partheno genetic algorithm for the binary quadratic programming problem, *Proc. of the Congress on Evolutionary Computation*, no.1, pp.356-363, 2001.

[19] M. J. Li, S. S. Fan and A. Luo, A Partheno-genetic algorithm for combinatorial optimization, *Neural Information Processing*, no.3316, pp.224-229, 2004.

[20] C. Darwin, *On the Origin of Species by Means of Natural Selection*, John Murry, London, 1859.

[21] H. H. Rosenbrock, An automatic method for finding the greatest or least value of a function, *The Computer Journal*, vol.3, pp.175-184, 1960.