A Fast Genetic Algorithm for Solving the Maximum Clique Problem

Suqi Zhang
Tianjin University of Commerce
School of Information Engineering
Tianjin, China

Jing Wang
Hebei University of Technology
School of Computer Science and Software
Tianjin, China

Qing Wu
Hebei University of Technology
School of Computer Science and Software
Tianjin, China

Jin Zhan
Tianjin University
School of Computer Science and Technology
Tianjin, China

Abstract—**Aiming at the defects of Genetic Algorithm (GA) for solving the Maximum Clique Problem (MCP) in more complicated, long-running and poor generality, a fast genetic algorithm (FGA) is proposed in this paper. A new chromosome repair method on the degree, elitist selection based on random repairing, uniform crossover and inversion mutation are adopted in the new algorithm. These components can speed up the search and effectively prevent the algorithm from trapping into the local optimum. The algorithm was tested on DIMACS benchmark graphs. Experimental results show that FGA has better performance and high generality.**

Keywords-the maximum clique; genetic algorithm; elitist selection; uniform crossover; inversion mutation; chromosome repair

I. Introduction

The maximum clique problem (MCP) is a typical combination optimal problem which has been proven to be NP-complete [1]. In recent years, a number of the heuristic algorithms based on genetic algorithm (GA) for solving the MCP have made great progress. The main algorithms include HGA [2], GENE [3], HSSGA [4] and EA/G [5]. Marchiori proposed HGA [2] and GENE [3], both of them consist of a simple genetic algorithm and a local search algorithm. HSSGA [4] was put forward by Singh, this algorithm consists of two parts: one for generating complete subgraphs by the steady-state genetic algorithm and one for extending complete subgraphs into cliques. In the EA/G [5], Zhang Qingfu et al. proposed a new guided mutation operator, the operator generates offspring through a combination of global statistical information and the local information in the solutions found so far. Harsh Bhasin also proposed a new genetic algorithm [6] in 2012, the crossover and mutation operator respectively are the single point crossover and single point mutation, no selection operator in the algorithm. In 2013, Harsh Bhasin improved the fitness function which proposed in [6] and applied roulette wheel selection in hybrid genetic algorithm [7], but no experimental results were given in the two articles. In addition, some novel and efficient algorithms were proposed, such as the [8], [9], and [10]. Wu Donghui [8] et al. induced the probability model to guide mutation when generating offspring and combined heuristic local search to find the maximum clique. In 2009, Zhou Benda et al. proposed an immune GA based on uniform design sampling for solving the MCP [9]. In 2011, Hu Nengfa et al. put forward a new chromosome evaluation method with multiple fitness functions in the [10].

So far, the main problems of genetic algorithm for solving the MCP are long-running and poor generality, aiming at these problems a fast genetic algorithm (FGA) is proposed in this paper. This algorithm was tested on DIMACS benchmark graphs. The results show the performance of FGA is very satisfactory both in terms of running time and generality.

II. The Maximum Clique Problem

A. Basic Definitions

Definition 1. Called $U=(V',E')$ is the complete subgraph of undirected graph $G=(V,E)$ if and only if $V'\in V,\ E'\in E$ and for $\forall u,v\in V'$ have $(u,v)\in E'$ establishment.

Definition 2. A complete subgraph U of undirected graph G is called a clique of G if and only if U do not contain in any other complete subgraph of G.

Definition 3. Called C is the maximum clique of undirected graph G if and only if C is a clique having the maximum number of vertices.

B. Mathematical Description of the Maximum Clique Problem

MCP is a special 0-1 programming problem and is described as follow:

Let $t:(0,1)^n\rightarrow 2^v$, $t(x)=\{i\in V: x_i=1\}$, $\forall x\in\{0,1\}^n$, $\forall S\in 2^v$, then $x=t^{-1}(S)=\{x_i: i=1,2,\ldots,n\}$, $x_i=\begin{cases}1, & i\in S\\ 0, & i\notin S\end{cases}$, n is the number of vertices in the graph G.

$$\max f(x)=\sum_{i=1}^{n}x_i \tag{1}$$

$$\text{s.t. } x_i+x_j\le 1,\ \forall(i,j)\notin E,\ \forall x\in\{0,1\}^n.$$

Tianjin Application Foundation and Advanced Technology Research Program （13JCZDJC26300）.

If x^* is the optimal solution of equation (1), then the set of $C = t(x^*)$ is the maximum clique of the graph G with $|C| = f(x^*)$.

III. Fast Genetic Algorithm for Solving MCP

A. Encoding and Fitness Function

Binary encoding is adopted in FGA, the chromosome consists of a string of bits (0 or 1), for example $X = (x_1, x_2, \cdots, x_i, \cdots, x_n), x_i = 0 \, or \, 1$, the chromosome X represents a subgraph of the graph G with n vertices, $x_i = 1$ (resp. $x_i = 0$) means that the vertex i is (resp. is not) in the subgraph.

Set the fitness function as follow:

$$f(X) = \begin{cases} \sum_{i=1}^{n} x_i & \text{chromosome X is a clique.} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The fitness of a chromosome is equal to the size of the subgraph it represents (i.e., the number of 1's occurring in the string) if this is a clique, and it is equal to zero, otherwise.

B. Chromosome Repairing Method Based on the Degree

The chromosome is not guaranteed to remain a clique after mutation and crossover operator are applied, so it is necessary to repair chromosome into a clique. In this section, a chromosome repairing method based on the degree is introduced. The proposed repairing method consists of two steps: the first step is extracting a complete subgraph from the subgraph; the second step is extending the complete subgraph into a clique.

The chromosome $X_j = (x_1, x_2, \cdots, x_i, \cdots x_n)$ represents a subgraph $G_j = (V_j, E_j)$ of the graph $G = (V, E)$, where $V_j \subset V$ and $E_j = (V_j \times V_j) \cap E$. The process of repairing the subgraph G_j to a clique is described as follow:

Extract a complete subgraph:

1. Delete the vertex with the smallest degree in G_j;

2. Update G_j;

3. Do step 1, 2 until G_j is a complete subgraph.

Extend complete subgraph into a clique:

1. Find the vertex candidate set PA of the complete subgraph G_j;

$$PA = \left\{ v \middle| v \in (V \setminus V_j), \forall u \in V_j, (u, v) \in E \right\}$$

2. Select the vertex with the maximum degree in PA, add the vertex into G_j;

3. Update $G_j = (V_j, E_j)$;

4. Repeat 1、2、3 until PA is empty, then get a clique $G_j = (V_j, E_j)$.

At present, most repairing methods used in the genetic algorithms are based on randomly choosing vertex to add into or delete from the current subgraph, such as in [2] and [3]. They thought random choice could prevent population from premature and trapping into local optimum, but in [11], a large number of experimental results showed the direct ratio correlation between the degree of the vertices and their frequency of being randomly chosen in local search. Therefore randomly choosing vertices in PA for many times and choosing based on the degree of the vertex will not make quite difference in the results, but the method based on the degree can eliminate short-term randomness and improve searching speed.

C. Elitist Selection

Elitist selection operator works at each family not at the population. For every mating pair, two offspring are created and the four individuals form the family, then the best two in the family are chosen into the next generation. The process of elitist selection is described as follow:

1. Shuffle the population;
2. Randomly pair every individual in the population;
3. Apply crossover operator with the probability of p_c on every mating pair to generate two offspring, and then form the family;
4. Apply mutation operator with the probability of p_m on the two offspring in every family;
5. Choose the best two individuals of each family into the next generation.

Elitist selection can ensure the quality of the offspring population is not lower than the parent population and the algorithm continuous convergence. Most algorithms used roulette wheel selection to choose the mating pair in elitist selection. The roulette wheel selection makes that the chromosomes with greater fitness value are repeatedly chosen, then the total fitness of the offspring population will be increased more quickly, but at the same time the diversity of the offspring population will be decreased. At last, the algorithm will fall into local optimum more easily. In this paper, elitist selection begins with random pairing, does not consider the fitness of the individual and makes every individual chosen with the same probability, so the diversity of the population is better.

D. Uniform Crossover and Inversion Mutation

In this paper, uniform crossover [12] operator is applied which is the same as the crossover operator in [3], so the detail will be not mentioned in this paper. Mutation operator used in FGA is inversion mutation whose mutation range is wider than swap and signal mutation, it has not been used in other literatures.

The concrete process of inversion mutation involves two major steps: (1) From 1 to the length of the individual, randomly assign two positions as inversion points; (2) Reverse the string between two inversion points. Figure 1 is a sample of inversion mutation in binary encoding individual, the inversion points are 4 and 8.

Inversion mutation

110 **01001** 10 $\longrightarrow$ 110 **10010** 10

Figure 1. Inversion mutation

The chromosome repairing method based on the degree can accelerate the process of repairing greatly. At the same time, it will make the population trap into the single mode more easily. In order to increase the diversity of the population, it is necessary to employ uniform crossover and inversion mutation that have wider range than other operators. In FGA, the configuration plays a key role in improving the speed and preventing the algorithm from trapping in local optimal solution.

E. Algorithm Process

The procedure of the FGA algorithm is described as follow:

1. Randomly generate initial population $P(t)(t=1)$;

2. Use the repairing method based on the degree to repair every individual in $P(t)$;

3. $t=t+1$;

4. Randomly pair every individual in $P(t-1)$;

5. Apply uniform crossover with the probability of p_c on every mating pair to generate two offspring;

6. Apply inversion mutation with the probability of p_m on the two offspring;

7. Use the repairing method based on the degree to repair the two offspring and calculate their fitness;

8. Choose the best two individuals of each family into the next generation to form $P(t)$;

9. Do step 3, 4, 5, 6, 7 and 8 until reaching the maximum number of generations or the optimal solution;

10. End.

IV. Experimental Result

The algorithm in this paper was tested on the set of well-known benchmark instances consisting of randomly generated graphs with known maximum clique and of graphs derived from various practical applications. The algorithm was run on the matlab2010a with Intel(R) Core(TM) i7-2600CPU@ 3.40GHz processor. Firstly, the experimental results of FGA with different parameters are given. Secondly, FGA is compared with the best four evolutionary approaches— HGA, GENE, HSSGA and EA/G. Lastly, time efficiency is compared between FGA and GENE.

A. Parameters Setting of FGA

With the size of population is 50, the maximum number of generations is 100, crossover probability and the mutation probability parameter are set on 1 and 0.1 respectively (i.e. $p_c=1$, $p_m=0.1$), FGA is run 100 times for each graph and the experimental results are shown in Table Ⅰ. The first three columns indicate the name, the size (i.e. the number of nodes) and the density of the graph. Next column indicates the average and the best size of the clique generated by FGA. The last column contains the best clique size known from the references [2-5].

Table Ⅰ shows that the FGA has obtained the known optimal solutions of 24 graphs in the 34 benchmark graphs, 10 of them are near optimal solution. By changing the size of population to 100, other 7 graphs can obtain the best solutions which are shown in Table Ⅱ. Yet, graphs C2000.9, C4000.5 and keller6 can not reach the optimal solutions (other GA algorithms also cannot reach).

TABLE I. Experience Results of FGA (Population Size=50)

Name	Nodes	Density	Clique Max	Clique Avg	BR
c125.9	125	0.898	34	34.0	34
c250.9	250	0.898	44	43.9	44
C500.9	500	0.900	56	53.8	57
C1000.9	1000	0.901	65	62.6	67
C2000.5	2000	0.500	16	15.1	16
C2000.9	2000	0.900	74	71.1	78
C4000.5	4000	0.500	17	16.2	18
DSJC500_5	500	0.501	13	12.7	13
DSJC1000_5	1000	0.500	15	14.0	15
brock200_2	200	0.496	12	10.7	12
brock200_4	200	0.657	16	15.8	17
brock400_2	400	0.749	25	24.0	29
brock400_4	400	0.748	25	23.9	33
brock800_2	800	0.651	21	19.5	21
brock800_4	800	0.649	20	19.3	21
gen200_p0.9_44	200	0.900	44	42.1	44
gen200_p0.9_55	200	0.900	55	55.0	55
gen400_p0.9_55	400	0.900	55	51.2	55
gen400_p0.9_65	400	0.900	65	65.0	65
gen400_p0.9_75	400	0.900	75	75.0	75
hamming8-4	256	0.639	16	16.0	16
hamming10-4	1024	0.828	40	40.0	40
Keller4	171	0.649	11	11.0	11
Keller5	776	0.751	27	26.4	27
Keller6	3361	0.818	57	54.7	59
p_hat300-1	300	0.243	8	8.0	8
p_hat300-2	300	0.488	25	25.0	25
p_hat300-3	300	0.744	36	36.0	36
p_hat700-1	700	0.249	11	10.7	11
p_hat700-2	700	0.497	44	44.0	44
p_hat700-3	700	0.748	62	62.0	62

p_hat1500-1	1500	0.253	11 11.0	12
p_hat1500-2	1500	0.506	65 65.0	65
p_hat1500-3	1500	0.753	94 93.2	94

TABLE II. EXPERIENCE RESULTS OF FGA (POPULATION SIZE=100)

Name	Nodes	Density	Clique size		BR
			Max	Avg	
C500.9	500	0.900	57.0	54.2	57
C1000.9	1000	0.901	67.0	63.2	67
brock200_4	200	0.657	17.0	16.0	17
brock400_2	400	0.749	29.0	24.2	29
brock400_4	400	0.748	33.0	24.4	33
brock800_4	800	0.649	21.0	19.7	21
p_hat1500-1	1500	0.253	12.0	11.0	12

B. FGA Compared with Other Algorithms

In this section, FGA is compared with HGA, GENE, HSSGA and EA/G. The parameters setting of every algorithm are taken from [2], [3], [4] and [5] respectively and are shown in Table Ⅲ. EA/G does not need the crossover probability and mutation probability. The experimental results are shown in Table Ⅳ and Table Ⅴ.

Table Ⅵ compares performance of FGA with HGA, GENE, HSSGA and EA/G in terms of the number of graphs on which FGA found better results and the number of graphs on which FGA found worse results out of all 34 DIMACS benchmark graphs. This comparison is done for average clique size as well as best clique size. Table Ⅵ clearly shows the advantage of FGA. As shown in Table Ⅵ, HGA has better results than GENE, HSSGA and EA/G at the best clique size, though the number of average clique size is less than HSSGA and EA/G.

TABLE III. THE PARAMETERS OF FGA, HGA, GENE, HSSGA, EA/G

	FGA	HGA	GENE	HSSGA	EA/G
size of population	100	50	10	50	10
number of generations	100	100	2000	20000	2000
crossover probability(p_c)	1	0.8	0.9	0.8	—
mutation probability(p_m)	0.1	0.1	0.1	0.01	—

TABLE IV. THE BEST CLIQUE AND AVERAGE CLIQUE ON 34 DIMACS BENCHMARK OF FGA, HSSGA AND EA/G

name	FGA		HSSGA		EA/G		BR
	Max	Avg	Max	Avg	Max	Avg	
c125.9	34	34.0	34	34.0	34	34.0	34
c250.9	44	43.4	44	43.8	44	44.0	44
C500.9	**57**	54.2	56	54.2	56	55.2	57
C1000.9	67	63.2	66	64.1	67	64.4	67
C2000.5	16	15.1	16	15.4	16	14.9	16
C2000.9	74	71.1	74	71.0	72	70.9	78
C4000.5	17	16.2	17	16.8	17	16.1	18
DSJC500_5	13	12.8	13	13.0	13	13.0	13
DSJC1000_5	15	14.1	15	14.7	15	14.5	15
brock200_2	12	11.2	12	12.0	12	12.0	12
brock200_4	17	16.0	17	16.7	17	16.5	17
brock400_2	29	24.2	29	25.1	25	24.7	29
brock400_4	33	24.4	33	27.0	33	25.1	33
brock800_2	21	19.5	21	20.7	21	20.1	21
brock800_4	21	19.7	21	20.1	21	19.9	21
gen200_p0.9_44	44	42.8	44	43.1	44	44.0	44
gen200_p0.9_55	55	55.0	55	55.0	55	55.0	55
gen400_p0.9_55	55	50.6	53	51.4	55	51.8	55
gen400_p0.9_65	65	64.4	65	63.8	65	65.0	65
gen400_p0.9_75	75	75.0	75	75.0	75	75.0	75
hamming8-4	16	16.0	16	16.0	16	16.0	16
hamming10-4	40	38.7	40	39.0	40	39.8	40
Keller4	11	11.0	11	11.0	11	11.0	11
Keller5	27	26.3	27	26.9	27	26.9	27
Keller6	57	54.7	57	54.2	56	53.4	59
p_hat300-1	8	8.0	8	8.0	8	8.0	8
p_hat300-2	25	25.0	25	25.0	25	25.0	25
p_hat300-3	36	35.9	36	35.9	36	36.0	36
p_hat700-1	11	10.8	11	11.0	11	11.0	11
p_hat700-2	44	44.0	44	44.0	44	44.0	44
p_hat700-3	62	61.9	62	61.7	62	62.0	62
p_hat1500-1	12	11.0	12	11.5	12	11.1	12
p_hat1500-2	65	64.7	65	64.9	65	65.0	65
p_hat1500-3	94	93.0	94	93.1	94	93.7	94

TABLE V. THE BEST CLIQUE AND AVERAGE CLIQUE ON 34 DIMACS BENCHMARK OF FGA, HGA AND GENE

Name	FGA		HGA		GENE		BR
	Max	Avg	Max	Avg	Max	Avg	
c125.9	34	34.0	34	34.0	34	33.8	34
c250.9	44	43.4	44	42.6	44	42.8	44
C500.9	**57**	54.2	55	52.9	56	52.2	57
C1000.9	**67**	63.2	64	58.0	66	61.6	67
C2000.5	16	15.1	16	14.4	15	14.2	16
C2000.9	**74**	71.1	69	67.1	72	68.2	78
C4000.5	**17**	16.2	16	15.4	16	15.4	18
DSJC500_5	13	12.8	13	12.3	13	12.2	13
DSJC1000_5	15	14.1	15	13.7	14	13.3	15
brock200_2	12	11.2	12	11.6	12	10.5	12
brock200_4	17	16.0	17	15.6	16	15.4	17
brock400_2	29	24.2	29	23.5	24	22.5	29
brock400_4	33	24.4	33	24.1	25	23.6	33
brock800_2	21	19.5	21	18.8	20	19.3	21
brock800_4	**21**	19.7	20	18.7	20	18.9	21
gen200_p0.9_44	44	42.8	44	40.7	44	39.7	44
gen200_p0.9_55	55	55.0	55	55.0	55	50.8	55
gen400_p0.9_55	55	50.6	52	49.0	55	49.7	55
gen400_p0.9_65	65	64.4	65	55.8	65	53.7	65
gen400_p0.9_75	75	75.0	75	65.0	75	60.2	75
hamming8-4	16	16.0	16	16.0	16	16.0	16
hamming10-4	40	38.7	40	37.8	40	37.7	40
Keller4	11	11.0	11	11.0	11	11.0	11
Keller5	27	26.3	27	26.3	27	26.0	27
Keller6	**57**	54.7	53	51.4	55	51.8	59
p_hat300-1	8	8.0	8	8.0	8	8.0	8
p_hat300-2	25	25.0	25	25.0	25	25.0	25

p_hat300-3	36 35.9	36 35.2	36 34.6	36
p_hat700-1	11 10.8	11 10.3	11 9.8	11
p_hat700-2	44 44.0	44 43.9	44 43.5	44
p_hat700-3	62 61.9	62 61.2	62 60.4	62
p_hat1500-1	12 11.0	11 10.4	11 10.8	12
p_hat1500-2	65 64.7	65 64.7	65 63.8	65
p_hat1500-3	94 93.0	94 91.4	94 92.4	94

TABLE VI. A SUMMARY OF TEST RESULTS IN TABLE 4 AND TABLE 5

	FGA Best clique size		FGA Average clique size	
	Better	Worse	Better	Worse
HGA	8	0	25	1
GENE	13	0	30	0
HSSGA	3	0	4	20
EA/G	4	0	4	22

C. *Time Efficiency of FGA*

The run time of GENE is better than HSSGA and EA/G that is mentioned in [3], [4]. In order to prove the time efficiency of FGA, the comparing between FGA and GENE is done in this section. The parameters of size of population, maximum number of generations and running times for every graph are all set to 100. The shortest time and the average time to get the optimal solution are recorded in Table Ⅶ. The experimental results show that FGA has more efficiency than GENE on the running time.

TABLE VII. THE COMPARISON OF TIME EFFICIENCY OF FGA AND GENE

Name	FGA Time(s)		GENE Time(s)	
	Min	Avg	Min	Avg
c125.9	0.46	0.78	1.83	3.14
c250.9	2.45	9.19	10.12	21.55
C500.9	37.04	39.12	65.76	65.86
C1000.9	88.69	90.92	173.36	173.73
C2000.5	481.55	505.34	502.13	503.39
C2000.9	547.00	560.57	548.37	549.19
C4000.5	4289.62	4385.49	2307.98	2404.38
DSJC500_5	15.22	15.98	60.38	60.54
DSJC1000_5	14.03	42.43	104.10	154.91
brock200_2	0.86	5.02	3.11	17.69
brock200_4	0.80	8.89	21.53	22.51
brock400_2	6.96	18.09	47.22	47.93
brock400_4	4.12	17.50	47.17	47.87
brock800_2	21.48	35.94	115.98	116.41
brock800_4	21.81	36.25	115.77	116.17
gen200_p0.9_44	2.11	5.02	11.46	17.43
gen200_p0.9_55	0.96	1.57	2.95	4.47
gen400_p0.9_55	6.13	22.80	49.30	49.66
gen400_p0.9_65	5.92	8.24	31.66	44.86
gen400_p0.9_75	4.87	6.22	15.54	21.67
hamming8-4	1.28	1.31	0.58	3.20
hamming10-4	46.49	53.97	167.12	167.35
Keller4	0.60	0.63	0.54	1.93
Keller5	19.70	29.37	48.68	95.07
Keller6	2568.27	2598.18	1483.90	1507.54
p_hat300-1	1.70	1.78	2.68	7.01
p_hat300-2	1.82	1.93	5.46	9.01
p_hat300-3	1.86	3.36	11.35	22.33
p_hat700-1	13.12	14.72	19.14	57.90
p_hat700-2	13.64	15.10	20.49	24.08
p_hat700-3	15.56	21.74	67.81	102.09
p_hat1500-1	194.37	200.22	294.19	296.14
p_hat1500-2	241.58	254.99	56.09	163.82
p_hat1500-3	190.04	246.36	346.57	347.31

V. CONCLUSION

In this paper, FGA employs a chromosome repairing method based on the degree to speed up the local search, adopts elitist selection with random pairing, uniform crossover and inversion mutation to increase the diversity of population. The algorithm can balance the exploitation and exploration in the process. FGA algorithm is composed of the basic genetic operators, is not add other diversified or optimum selection process. Experience results prove its running time is shorter than other methods, and the algorithm for most of the graph can obtain the optimal solution, so it has better performance and high generality.

REFERENCES

[1] Wu Qinghua, Hao Jinkao. An adaptive multistart tabu search approach to solve the maximum clique problem. Journal of Combinatorial Optimization, 2013, 26(1): 86-108.

[2] MARCHIORI E. A simple heuristic based genetic algorithm for the maximum clique problem//SAC. 1998: 366-373.

[3] MARCHIORI E. Genetic, iterated and multistart local search for the maximum clique problem//Applications of Evolutionary Computing. Springer Berlin Heidelberg, 2002: 112-121.

[4] SINGH A and GUPT A K. A hybrid heuristic for the maximum clique problem. Journal of Heuristics, 2006, 12(1-2): 5-22.

[5] ZHANG Q, SUN J and Tsang E. An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE transactions on evolutionary computation, 2005, 9(2): 192-200.

[6] BHASN H and MAHAJAN R. Genetic algorithms based solution to maximum clique problem. International Journal on Computer Science and Engineering (IJCSE) , 2012, 4(8): 1443-1448.

[7] BHASIN H, KUMAR N and MUNJAL D. Hybrid genetic algorithm for maximum clique problem. International Journal.2013, 2(4): 2319-4847.

[8] WU Donghui, MA Liang. Improved genetic algorithm for maximum clique problem. Journal of Computer Application, 2008, 28(12): 3072-3073.

[9] ZHOU Benda, YUE Qin, CHEN Minghua. Immune genetic algo-rithm based on uniform design sampling for solving max-imum clique problem. Computer Engineering, 2010, 36(18): 229-231.

[10] HU Nengfa, TANG Weiping._Study on genetic algorithm for solving the maximum clique problem. Journal of Hubei University (natural science), 2011, 33(2): 256-259.

[11] PULLAN W, HOOS H H. Dynamic local search for the maximum clique problem. Journal of Artificial Intelligence Research, 2006, 25: 159-185.

[12] GE Jike, QIU Yuhui, WU Chunming, PU Guolin. Summary of genetic algorithms research. Application Research of Computers, 2008, 25(10): 2911.