A Study on the Traveling Salesman Problem using Genetic Algorithms

Zhigang Zhao [1], Yingqian Chen [1], Zhifang Zhao [2]

[1] *School of New Media,* Zhejiang University of Media and Communications, Hangzhou Zhejiang, China

[2] College of Civil Engineering and Architecture, Zhejiang University of Technology, Hangzhou Zhejiang, China

***Abstract* — Traveling Salesman Problem (TSP) is widely applied in various fields and can also be used to evaluate the merits of an algorithm, where there are many applications in real life situations that can be transformed into a TSP problem. Thus solving the problem has both theoretical and practical significance. Many methods can be used to solve TSP problems, such as branch limit method, dynamic programming, ant colony algorithm, etc. In this paper, we consider TSP problems based on genetic algorithms, as follows: i) we explain the principles of genetic algorithm, ii) we write C++ program to solve the TSP problem using genetic algorithms, iii) we analyze factors such as population size and crossover rate influence on the results, and iv) make comparisons between ant colony algorithms and genetic algorithms. Our results demonstrate that genetic algorithms used to solve TSP problems have good advantage in certain circumstances.**

Keywords-TSP, Genetic algorithm, C++, Ant colony algorithm

I. INTRODUCTION

The traveling salesman problem, also commonly known as the TSP, is a typical computer science problem. The problem itself is a quite simple concept: the salesman must visit each of all the cities within a certain scope for once and once only and return to the starting point ultimately [1]. As we all know, it is quite unlikely that we could find the precise optimal solution to the TSP problem, because the function of its path number is an exponential function. Therefore, it is vital to find the algorithm that could quickly yield the approximate value.

Over the last decades, there had been several approximation optimization algorithms with regard to this problem, such as the Tabu Search Algorithm TS [2, 3], the Ant Colony Algorithm [4], the Simulated Annealing Algorithm [5], the Artificial Neural Network [6], the LK Algorithm [7] and the Particle Swarm Optimization [8]. Due to the fact that the genetic algorithm has a strong global searching ability, it has become the research hotspot in recent years [9, 10]. Therefore, utilizing genetic algorithm to solve TSP is a relatively better method to obtain an approximate solution, as well as the direction with most potential, along which studying and developing this problem could result in a better solution to this problem.

II. MAJOR ALGORITHMS TO SOLVE THE TSP PROBLEM

A. Dynamic Programming Method

For a problem difficult to be solved, we usually break it down into multiple minor problems, and then by solving the minor problems, we could find the solution to the original problem. This is the basic idea of the dynamic programming method. If this algorithm is adopted to solve the TSP problem, its process is as follows: Assume that the shortest loop among three cities is to be S0—>S1—>S2—>S0, then:

Length (general loop) = length (S0—>S1) + length(S1—>S2—>S0), then the general loop Length is broken down into:

Length (general loop) = Min {length (0—>1)+ length(1—>…—>0),Length(0—>2)+ length(2—>…—>0)}

A standardized expression of the above formula would be:

$d(i,V) = \min\{C_{ik} + d(k, V-\{k\})\ (k \in V)$ （1）

$d(k,\{\}) = C_{ik}\ (k \neq i)$ （2）

In which, d (i, V) refers to the shortest distance of one starting from i and visiting each of all the points in the city cluster V for once and once only and then returning to the starting point. C_{ik} is the distance from i to k. Then the solution can ultimately be obtained by iteration in accordance with the formula. This algorithm belongs to the recursive algorithm, the time complexity of which is $O(n^2 2^n)$. Therefore, with the expanding of the problem magnitude, the needed space and time expenditures would also increase steadily, which causes the method to be not generally adopted.

B. Ant Colony Algorithm

The Ant Colony Algorithm is a probability algorithm through which the optimal solution can be found on the map. In 1992, Doctor Marco Dorigo unintentionally found out the certain fixed route behaviors of ants in search of foods, and then came up with this algorithm. From the collective behaviors of ant colony, we can get following information: ants would choose the following direction of crawling based on the amount of pheromone left over on their crawling path. This delicate communication method between ants could help them find food in the shortest time period. At present the Ant Colony Algorithm is applied to the combination optimization, and its flow diagram is as Fig.1:

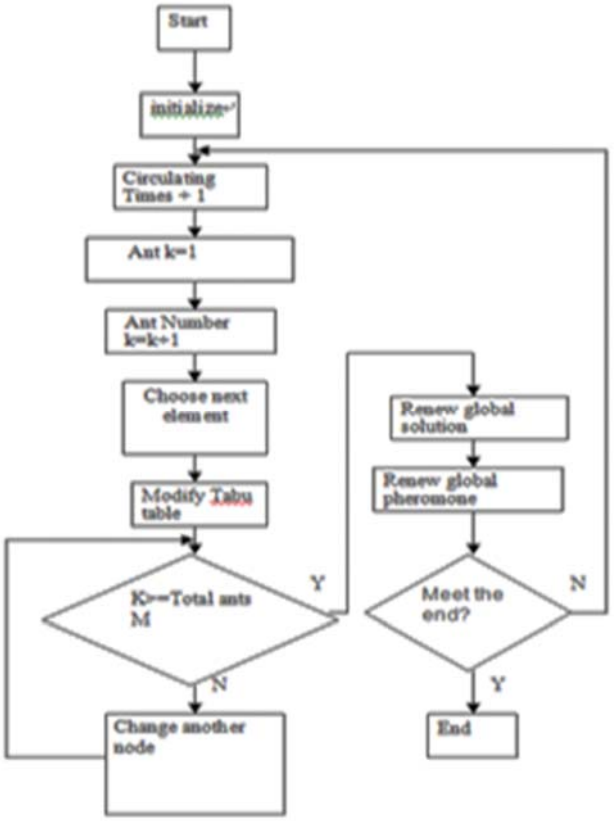

Figure 1. Flow diagram of ant colony algorithm

The Ant Colony Algorithm is a self-adaptive method with positive feedback. It has relatively less parameters, thus promoting the finding of optimal solution to the whole problem. Meanwhile, it is also prone to the problem of falling into the dilemma of local optimum.

B. Genetic Algorithm

The Genetic Algorithm is essentially a heuristic searching algorithm used to solve such problems as finding the optimal solution [11]. Therefore, to design a suitable objective function and genetic manipulations is a very important step. To solve the traveling salesman problem through the Genetic Algorithm generally has the following steps, the flow diagram of which is shown in Fig. 2 below:

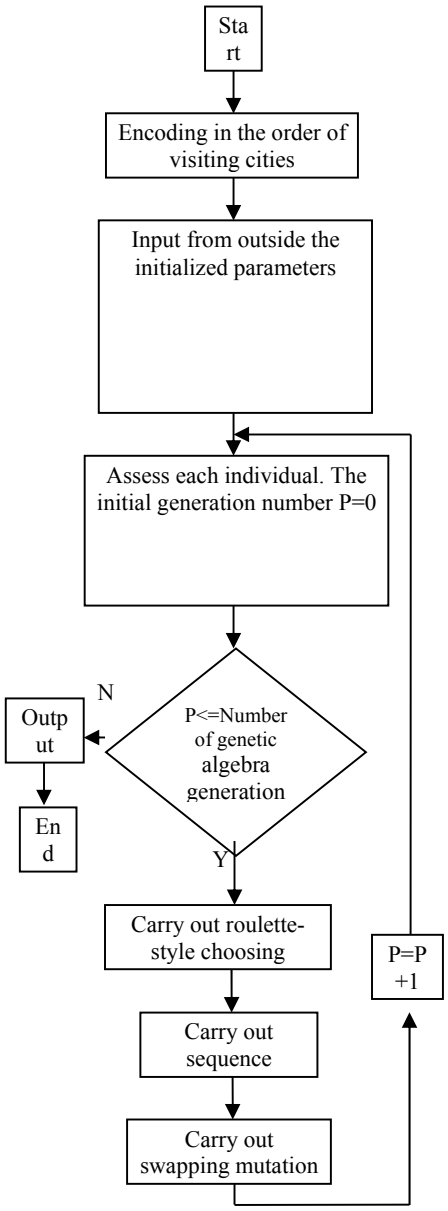

Figure 2. Flow diagram of genetic algorithm

III. III. SIMULATION AND RESULT ANALYSIS

A. System Simulation

The major parameters that need determining during the initialization of system include the city number, the population size, the mating rate, the mutation rate and the ultimate genetic algebra, etc.

Encoding is vital to the design of genetic algorithm. And in this case the encoding is done according to the route

arrangement of traversal sites. The numbers are identified according to the decimals, and every genome is a decimal numerical string. Assume that there are ten sites: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. If the traveling salesman visits in the order of (2, 6, 4, 3, 9, 7, 8, 1, 0, 5, 2), then the individual of the population could be encoded by utilizing (2, 6, 4, 3, 9, 7, 8, 1, 0, 5, 2) [12,13]. This is the most direct encoding method, and the primary fault with it is that after the crossover operation, it visits repeated routes, illegal routes. The fitness function is utilized to assess the surviving capacity of a certain individual in the population, which is also meant to reflect the survival of the fittest in Nature. In the TSP problem, the shorter route of every chromosome is, the higher probability of its survival; and hence the chromosomes with the short routes are chosen for manipulation. The paper employs the roulette algorithm. The crossover manipulation adopts the order crossover method. Mutation operation is used to improve the GA's searching ability in small area [14], and to protect the diversity of the population, thereby averting convergence ahead of schedule. The mutation adopted by the paper is swapping.

The example cited by this paper is the distance matrix composed of the 14 colleges or universities in the Xia Sha Higher Education Zone of Hangzhou. The distance matrix is map[14][14],just as Table 1.

TABLE 1 THE DISTANCE MATRIX

0	1.4	2.4	1.7	1.1	2.2	2.7	3.9	2.0	2.2	3.9	2.2	1.5	2.5
1.4	0	1.1	0.81	1.4	0.31	1.4	4.3	2.9	1.7	3.5	1.7	2.4	3.8
2.4	1.1	0	1.4	2.4	0.89	1.3	5.2	3.9	2.7	4.5	2.7	3.3	4.9
1.7	0.81	1.4	0	1.2	1.6	1.7	5.5	3.5	3.7	5.5	3.7	3.0	4.2
1.1	1.4	2.4	1.2	0	2.1	2.2	4.8	2.9	3.1	4.9	3.1	2.4	3.4
2.2	0.31	0.89	1.6	2.1	0	1.6	4.9	3.6	2.4	4.3	2.4	3.1	4.6
2.7	1.4	1.3	1.7	2.2	1.6	0	5.9	4.5	3.4	5.2	3.4	4.0	5.2
3.9	4.3	5.2	5.5	4.8	4.9	5.9	0	1.1	1.6	0.7	1.6	1.8	2.3
2.0	2.9	3.9	3.5	2.9	3.6	4.5	1.1	0	0.87	1.0	0.87	0.5	0.94
2.2	1.7	2.7	3.7	3.1	2.4	3.4	1.6	0.87	0	0.54	0.01	0.76	2.1
3.9	3.5	4.5	5.5	4.9	4.3	5.2	0.7	1.0	0.54	0	2.0	1.8	2.2
2.2	1.7	2.7	3.7	3.1	2.4	3.4	1.6	0.87	0.01	2.0	0	0.76	2.1
1.5	2.4	3.3	3.0	2.4	3.1	4.0	1.8	0.5	0.76	1.8	0.76	0	1.3
2.5	3.8	4.9	4.2	3.4	4.6	5.2	2.3	0.94	2.1	2.2	2.1	1.3	0

In theory, the size of initial population in the GA determines the diversity of the population. If the initial population is too small in size, it may give rise to the local optimum of the program with the approximation optimal solution being unable to be acquired. Although the local optimum can be avoided by increasing the size of population, yet the arithmetic speed of the program would be slowed down if the population is too large resulting further in the low efficiency of algorithm. The Fig.3 shows the situation changes in the optimal solution with change in the size of the population.

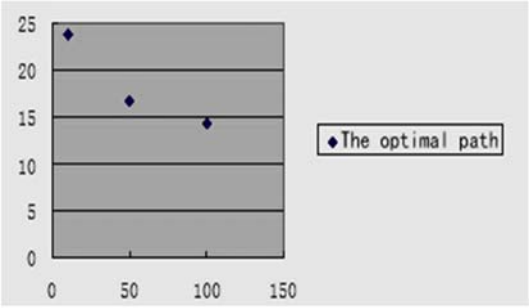

Figure 3.Optimal path of the different size of populations

In Table 2, pc (cross-over rate)=0.8pm (mutation rate)=0.05. The table is compare and contrast of experiment results between the initial population having a size of 10 and 100 respectively and the largest heredity generation number of 1000 and 2000.

TABLE 2 CONTRASTING TABLE OF THE ALGORITHM OUTCOME OF DIFFERENT INITIAL POPULATIONS

Number of Hereditary Generations	1000		2000	
Size of Population	Optimal Solution's Algebra	Optimal Solution	Optimal Solution's Algebra	Optimal Solution
10	400	23.72	1100	22.44
100	400	14.29	400	14.29

The above outcome evidently shows that when the initial population becomes larger, the solution obtained will be closer to the optimal solution. But it is not difficult to find that the circulating times increase with the expanding of the initial population, the result of which is the lowered efficiency in algorithm. The suitable initial population size should be set at between 50 and 100 for the TSP among the 14 colleges or universities.

In the meantime, the change in hereditary generation numbers will also change the optimal solution, which is shown in Fig.4.

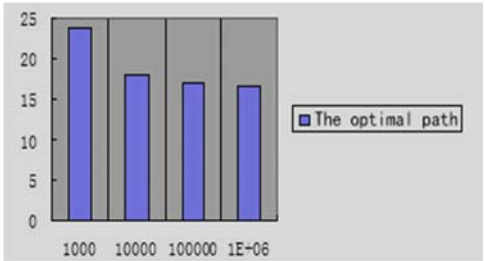

Figure 4. Chart of optimal paths of different hereditary generations

It can be easily seen from the experiment that the larger the hereditary generation numbers, the closer to optimal the outcome of the TSP is. But the exceeding large number of hereditary generations would cause the over large expenditure of computers. Therefore, in this TSP among the

14 locations, the hereditary generation number should better be set at between 1000 and 5000.

Crossover manipulation is an important method of GA producing new chromosomes. If the value is set too high, the previously fine gene segments in the chromosome would be destroyed, which is detrimental to the evolution. However, if the value is set too low, the convergence rate would be much slowed down. So the value of crossover rate is set through the experiment, the results of which are shown in the following Fig.5.

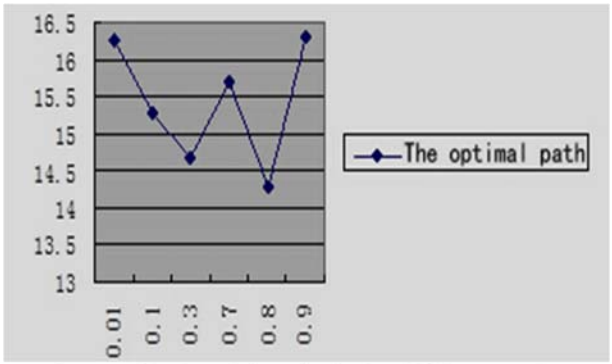

Figure 5. Line chart of the optimal paths of different pc values

When the mutation rate pm is relatively small, the mutation operators have the capacity to search in a small area, enabling the GA to accelerate its convergence to get the global optimal solution. But if it is excessively small, the ahead-of-time convergence would occur. Change the

mutation rate, and the results of optimal paths are as shown in Fig.6.

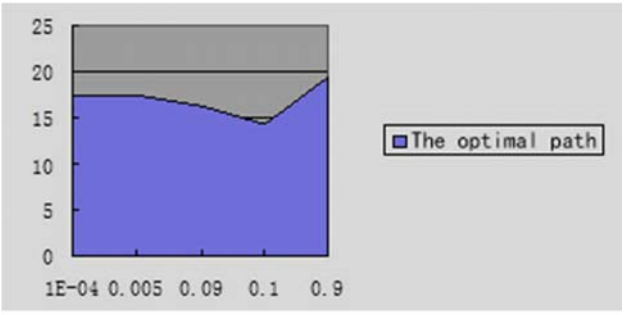

Figure 6 . Area graph of optimal paths of different pm values

Table 3 is the contrast of different results of the two groups by changing the crossover rate and mutation rate when the initial population is 10, As Table 3.

TABLE 3 CONTRASTING TABLE OF THE OUTCOME OF ALGORITHMS WITH DIFFERENT CROSSOVER RATES AND MUTATION RATES

Conditions	pc=0.9 pm=0.001				pc=0.5 pm=0.5			
Hereditary Generation Numbers	500		1000		500		1000	
Times of Arithmetic	Optimal Solution Algebra	Optimal Solution	Optimal Solution Algebra	Optimal Solution	Optimal Solution Algebra	Optimal Solution	Optimal Solution Algebra	Optimal Solution
1	300	24.64	300	24.64	300	21.72	700	21.10

From the above results, it can be seen that Status 1 sees the convergence at too early a stage. The Status 2 can get the optimal solution by increasing the size of the population, but Status 1 can hardly do so to break through the constraints of local convergence.

Through contrasting experiments, we know that through lowering the crossover rate pc and raising the mutation rate pm, we can reduce the probability of algorithm being mired into local convergence, while at the same time slowing down the algorithm's convergence. Only when the hereditary generation number is relatively increased can we more easily get the global optimal solution. In this paper, the pc should be between 0.4 and 0.9, while the pm should be between 0.001 and 0.1.

B. Contrast of the Genetic Algorithm and Ant Colony Algorithm

As shown in pictures from Fig.7 to Fig.9, the Ant Colony Algorithm has a faster convergence. When the 290th generation is reached, the result is 15.04, while the 1000th generation is approached, the result is 20.67, deviating from the optimal solution and wasting the expenditure of computers. When the 1000th generation is reached by the Genetic Algorithm, the optimal path is 14.29, the results of which are apparently superior to those of the Ant Colony Algorithm.

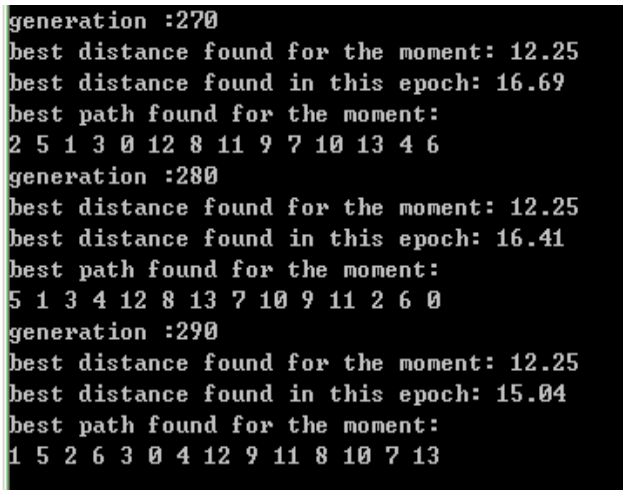

Figure 7. Picture of the results of the ant colony algorithm at the 290th generation

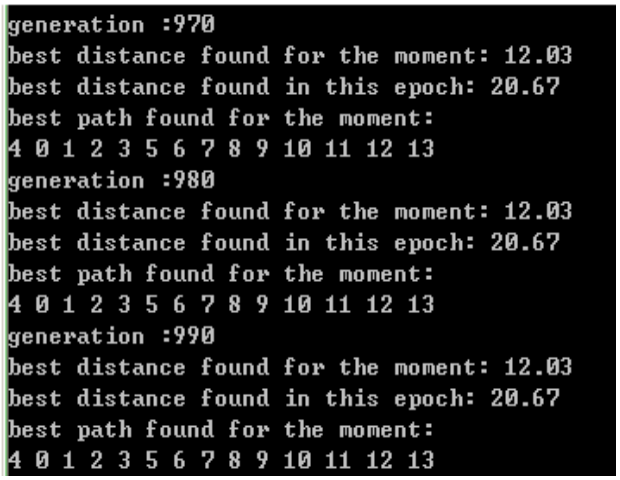

Figure 8.Picture of the results of the ant colony algorithm at the 990th generation

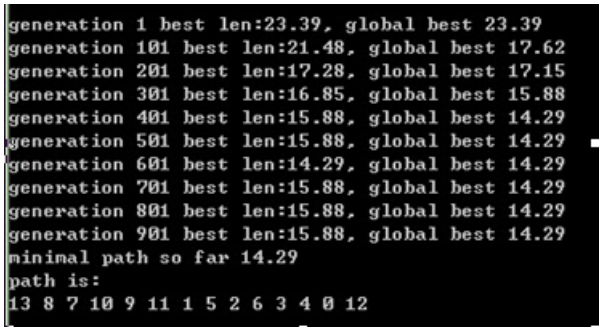

Figure9. Picture of the results of the genetic algorithm at the 900th generation

IV. CONCLUSION

The traveling salesman problem (TSP) is widely applied in different fields. It can not only be used to solve various practical problems, but also applied to verify the performance of algorithms. The Genetic Algorithm is a method of finding the optimal solution through simulating the hereditary phenomena in natural evolution, which has outstanding advantages in solving the combination optimization problems, like the TSP. The work that has completed by the author in this paper can be concluded in the following aspects:

(1) In the design of the Genetic Algorithms, the selecting manipulation adopts the roulette-type algorithm, the crossover manipulation chooses the sequence crossover method OX1, and the mutation manipulation employs the simplest method---swapping mutation.

(2) After the system simulation is realized, the following two conclusions are made through comparing and analyzing the effects of hereditary generation numbers, population size, and crossover probability pc and mutation rate pm on results:

The size of population decides the diversity of population. If the individuals are too small, the Genetic Algorithm would have a convergence ahead of time, resulting thus in that it falls into local optimum. Under such circumstances, the global optimum would hardly be reached even if the hereditary generation number is raised. If the population's individuals are too large, the genetic manipulation would have too many circulations, increasing the expenditure of computers and reducing the efficiency of algorithms. Hence, usually for the 14 locations, the size of population should better be set at between 50 and 100.

Reducing crossover rate pc and increasing mutation rate pm could prevent the algorithms from falling into local optimum as much as possible, while slowing down the convergence of the algorithms. The number of the hereditary generations should be comparatively raised to help us more easily get the global optimal solution. In the TSP problem, the pc should be set at between 0.4 and 0.9, while the pm should be at 0.001 to 0.1.

(3) At last, by analyzing the advantages and disadvantages of the Ant Colony Algorithm and the Genetic Algorithm in solving the TSP, we find that when the cities are fewer, the Genetic Algorithm's speed is higher, and the result is closer to the optimal one. When the locations are larger in number, the outcome of the Genetic Algorithm may be better, but the expenditure of computers is more. The Ant Colony Algorithm, on the contrary, can get an approximation solution within much fewer generations.

ACKNOWLEDGMENT

This work is supported by the Natural Science Foundation of China (Project No.71371169 and No.51479178), Zhejiang Provincial Natural Science Foundation of China (Project No. LY14E090006), and the Colleges and Universities Visiting Scholar Teacher Professional Development Program (FX2014069) of Zhejiang Province.

REFERENCES

[1] Wang Xiaoping, Cao Liming. Genetic Algorithm---Theory, Application and Software Realization . Xi'an: Xi'an Jiaotong University Press. 2002.1.

[2] Glover F.Tabu Search- Part I. ORSA Journal on Computing,1999,1(3): pp.190-206.

[3] Glover F..Tabu Search- Part II.ORSA Journal on Computing, 1999, 2(1): pp.4-32.

[4] ZHOU Y.Runtime Analysis of an ant colony optimization algorithms for TSP instance.IEEE Transactions on Evolutionary Computational,2009,13(5), pp.1083-1092.

[5] SONG C,LEE K.Extended simulated annealing for augmented TSP and multi-salesmen TSP.Don Wunsch,Michael Hasselmo,et al.Proceedings of the International Joint Conference on Neural Networks.Portland,Oregon:IEEE Neural Networks Society,2003, pp.2340-2343.

[6] ABDEL-MOETTY S.Traveling salesman problem using neural network technique.Ahmed Zoweil,Dokki,Giza..The 7th International Conference on Informatics and Systems.Cairo,Egypt:IEEE Conference Publications Program,2010, pp.1-6.

[7] LIN S,KERNIGHAN B.An effective heuristic algorithm for the traveling salesman problem.Operation Research,1993,21(2), pp.486-515.

[8] JAMES K,EBERHART R.A discrete binary version of the particle swarm algorithm.Proceeding of the IEEE International Conference on Systems,Man and Cybernetics,1997, 5, pp.4104-4108.

[9] Wu Bing, Shi Zhongzhi. A Segmentation Solving Algorithm of TSP Based on Ant Colony Algorithm . Chinese Journal of Computers, 2001, 24(12), pp.1228-1233.

[10] [10]Yang Hui, Kang Lishan, Chen Yuping. A Genetic Algorithm of Solving TSP Based on Constructing a Gene Pool . Chinese Journal of Computers, 2013, 26(12) pp.1753- 1758.

[11] C. F. Lima, M. Pelikan, D. E. Goldberg, F. G. Lobo, K. Sastry and M. Hauschild, Influence of selection and replacement strategies on linkage learning in BOA. In Proceedings of the 2007 Congress on Evolutionary Computation (CEC-2007), pp. 1083–1090. IEEE Press,2007.

[12] T. Villmann, B. Villmann and V. Slowik, Evolutionary algorithms with neighborhood cooperativeness according neural maps, Neurocomputing 2004, 57, pp. 151–169.

[13] M. Hauschild, M. Pelikan, K. Sastry, , and Goldberg, D. E. (2012). Using previous models to bias structural learning in the hierarchical BOA. Evolutionary Computation, 2012, 20(1):pp. 135–160.

[14] M. Hauschild, M. Pelikan, K. Sastry, and C. Lima, Analyzing Probabilistic Models in Hierarchical BOA. IEEE Transactions on Evolutionary Computation, 2009 , 13(6), pp. 1199–1217.