An Improved Genetic Algorithm for Solving the Traveling Salesman Problem

Peng Chen
Information Engineering
Guangdong Ji Dian Polytechnic
GuangZhou, China

***Abstract*—In this paper, on the basis of the original genetic algorithm, an improved genetic algorithm for the Traveling Salesman Problem (TSP) is proposed. Firstly, the diversity of species is ensured by amending the calculation method of the individual fitness. Secondly, the mutation operator is improved by the combination of shift mutation and insertion mutation. Before the crossover, the operator checks whether the degradation phenomenon will occur. Finally, experimental results further determine that above improvements provide a significant effect for solving the TSP.**

Keywords-Genetic Algorithm; Traveling Salesman Problem Solving; Optimization

I. INTRODUCTION

Genetic Algorithm (GA) is a computational model which simulates Darwinian genetic selection and biological evolution. GA was firstly presented by John Holland in 1975. The algorithm can be applied to get a satisfactory solution. As one of the traditional methods, it is often employed to solve combinatorial optimization, pattern recognition, image processing and other complex problems. The main advantages of GA are: simple, universal and robust [1]. However, relative to its distinct biological basis, the GA's imperfections in theoretical basis are widely recognized and mainly manifested in the absence of complete theoretical explanation for the mechanism of algorithm and the lack of an extensive and complete theory of its convergence. Deficiencies in performance of the algorithm have led to the further study of its theory and performance.

Traveling Salesman Problem (TSP) is a combinatorial problem, which has widely troubled different computer scientists. The issue is how to find the shortest path which has to pass all the cities if it is supposed the given coordinates of the cities. Main traditional methods include branch-and-bound method, successive correction, greedy algorithm, minimum-cost spanning tree (MST), local search, multilateral of adjustment, etc. Additionally, modern optimization methods include simulated annealing [9], genetic algorithm, ant algorithm [10], particle swarm optimization [11], tabu search algorithm [12], Hopfield neural networks [13] [14], etc. Amongst them, genetic algorithm is one of advanced technologies.

There is a wide range of applications for the solution of this problem. There is a high theoretical value on researching the TSP, which can be applied in logistics, traffic control, and other applications [3].

After this brief introduction and definition in Section 1, general process of GA system is reviewed in Section 2, followed by Section 3 describing the improvements of the GA, Section 4 presenting results and discussion, and Section 5 drawing the conclusion of this research.

II. THE GENERAL PROCESS OF BASIC GENETIC ALGORITHM

(1) Initialize the parameters: set the size of population as N. Set the evolution of generation as G. Set the probability of crossover as Pc. Set the probability of mutation as Pm.

(2) Set the initial value of G as 0 and randomly generate the initial population as P_0, number of which is N, then P0 = (x_1, x_2, x_n).

(3) Choose a calculation function of fitness, e.g., (f_i (x), (i = 1,2, ... N)) by which the fitness value of each organism is calculated. The fitness value can be employed to evaluate the performance of an organism in a group.

(4) Selection operator chooses two members of the present generation to participate in the next operations: crossover and mutation.

(5) Crossover operator and mutation operator produce a new generation, e.g., $P_{G+1,}$ depending on the chosen members in the previous step.

(6) Determine whether the group performance meets the target. If it does, algorithm stops running; otherwise the processing returns to step 3 [2].

Fig.1 depicts an overview of a simple Genetic Algorithm.

978-1-4673-4714-3/13/$31.00 ©2013 IEEE

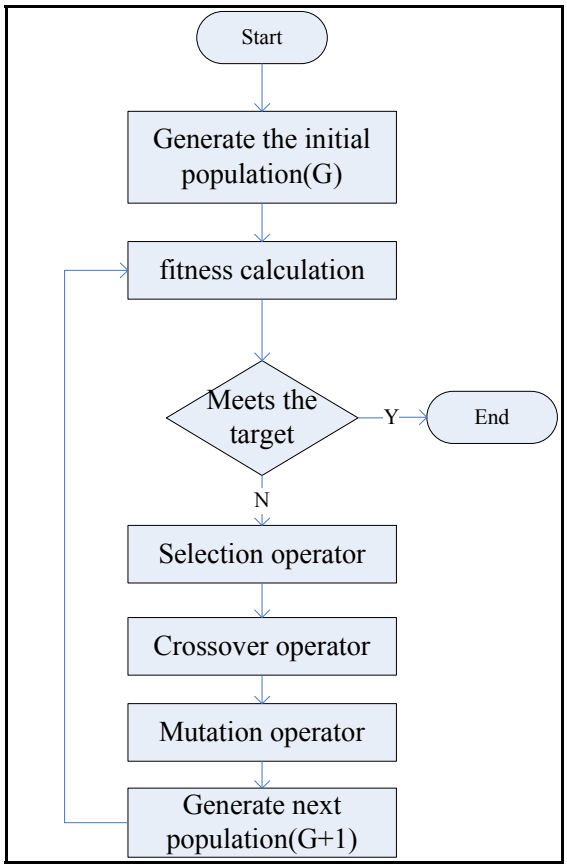


Figure 1. Genetic algorithm cycle

III. THE IMPROVEMENT OF GENETIC ALGORITHM

A. Genetic Code

In order to set the connection between the representation of genetic algorithm and the targeted problem, it is necessary to determine the encoding and decoding operator. The encoding mode determines the crossover operator. However, some experts proposed various coding methods. In GA, binary is generally applied to present target problems. According to the character of the TSP, each city can be represented with a decimal number such as 1, 2 , 3….. The decimal number of a city represents a path = (1, 2,... 10). Therefore solving the TSP means finding the shortest distance between any two numbers which represent two cities. In crossover operator and mutation operator, the decimal number can be directly changed because this representation can save encoding and decoding time (see Fig 2).

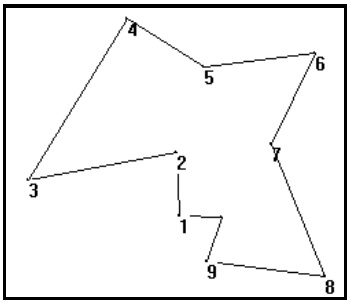


Figure 2. Encoding with decimal number of cities

B. The Fitness Function

A fitness function evaluation incorporated to assign a value to each organism is noted as f_i. High fitness evaluation means that the organisms have a high probability to be chosen. Selecting organisms based on fitness value is a major factor in GA. In TSP the fitness can be calculated by the following equation. The method first needs to calculate the total distance of all the cities in the sequence, then computes the maximum total distance of the present generation, and then uses the equation:

$$F(t) = \max - \left(\sum_{i=0}^{n-2} D(i, j) + D(0, n-1) \right) \quad (1)$$

n represents the number of the cities; flag max means the longest path length in the present generation. $D(i,j)$ means the distance between city i and city j. It can be seen from the formula that individual with a shorter path will get a high fitness points and the individual has a high genetic probability into the next generation. However, this method may cause convergence too fast and reduce the diversity of the individual.

The second method is to recalculate the rank of fitness value of each organism on the basis of the previous calculation shown in table I, and then select the organism by adaptive ratio. Because this method chooses the rank of fitness score, it is evenly distributed. This way avoids a large number of offspring produced from a small number of high adaptive organisms however it may result in slow convergence.

TABLE I. RANK OF FITNESS VALUE.

Cities	Fitness	Rank
12345678910	200	1
12657834910	100	3
15768314910	120	2

The third method needs to calculate the mean of all fitness values, then calculate the variance σ, then use the following equation:

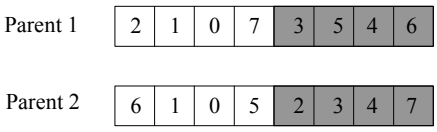
$$f(t) = \frac{\text{old fitness - average fitness}}{2\sigma} \quad (2)$$

$\sigma = \sqrt{\frac{\sum (f - a)^2}{N}}$, in which f means organism fitness value; a means average fitness of present generation; N means the size of population.

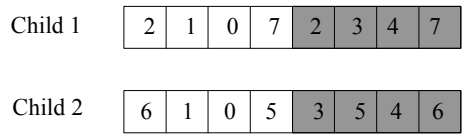
At the beginning of the algorithm, organism fitness is distributed uniformly. High fitness individual with a low probability is assigned to future generations. Therefore, the algorithm firstly ensures the diversity of the species. However, later when the individual is not very different, a better fitness organism will have a greater probability to be assigned to future generations. This will help algorithm in rapid convergence in the late stage.

C. Improved Crossover Strategy

Crossover and mutation operator ensures algorithm in global convergence. Crossover is to choose two different organisms to exchange some parts of genes to produce a new organism. The main character of the TSP is that the solutions must be a valid traveling path. Exchanging parts of two solutions will usually produce an invalid solution. For example, two selected random city numbers in parents represent two parents of length 8.



After swapping the data at the crossover point, we can get two children solution:



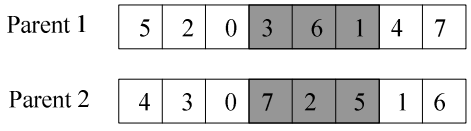
Both of the resulting children solutions are invalid because the number is repeated.

In order to eliminate the duplication researchers invented several new crossover operators in the TSP. They are: Partially Mapped Crossover (PMX), Order Crossover (OX), as described briefly below.

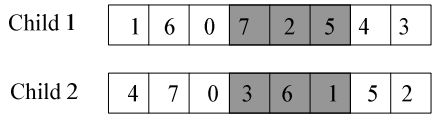
(1) Partially Mapped Crossover

- Select two positions along the string uniformly at random. The substrings defined by the two positions are called the mapping sections.
- Exchange two substrings between parents to produce proto-children.
- Determine the mapping relationship between two mapping sections.
- Legalize offspring with the mapping relationship.

For example, the mapping sections are (3,6,1) and (7,2,5) in the parents with length 8.



From the mapping sections, the relationship between parents can be determined. 3 corresponds with 7, 6 corresponds with 2, and 1 corresponds with 5. So exchanging the upper number can get the legalize offspring as follows.

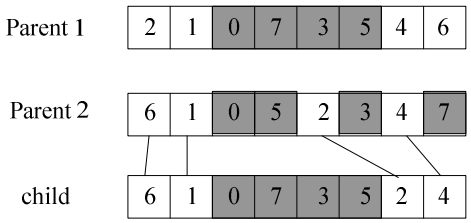


(2) Order Crossover

Order crossover is a kind of variation of PMX with a different repairing procedure.

- Select a substring from a parent at random.
- Produce a proto-child by copying the substring into the corresponding position of it.
- Delete the cities which are already in the substring from the second parents. The resulted sequence of cities contains the cities that the proto-child needs.
- Place the cities into the unfixed positions of the proto-child from left to right according to the order of the sequence to produce an offspring.

For example, select a substring at random as following （0,7,3,5）from parent 1. The substring will be directly copied into the corresponding position of the child. Delete cities of (0,5,3,7) from the second parent. Place cities of (6,1,2,4) into the child.



There is a variety of methods for crossover operators. These crossover operators base on the circumstances of each run and sometimes can not get a globally optimal solution. In order to prevent this phenomenon occurring, whether degeneration occurs has to be checked before crossover happens. If fitness score is lower than the mean of last generation, crossover operator will be stopped or the organism will be replaced by another good organism.

D. Improved Mutation Strategy

Experts presented different kinds of mutation operators in GA. The general methods are shift mutation, insertion mutation, inverted mutation, etc. The idea of shift mutation is to randomly remove a certain chromosome of an organism which is inserted into a random position. For example, suppose one organism expressed as (1,2,3,4,5,6,7,8,9,10). In accordance with the shift mutation, if the fetch length equals 2. After shift mutation operator organism can be expressed as (1,4,5,6,7,2,3,8,9,10). This operator allows genetic algorithm converge quickly to a shorter path. Insertion mutation is similar to the shift mutation. But Insertion mutation only removes one gene at random at one time and inserts it into another position. After insertion mutation operator organism can be expressed as (1, 2,4,5,6,7, 3,8,9,10). Experiments show that insertion mutation is better than any other mutation operator.

Here two mutations are combined. In the initial stage, insertion mutation is adopted in order to guarantee the diversity of the gene. After several generations of iteration (such as 100 generations), the same segment of organisms is found as a local minimum solution. These local minimum solutions can be chosen as shift mutation sub-segment. This method can ensure that excellent genes can be protected to the next generation.

From the above analysis, the main steps of the improved algorithm can be described as follows:

(1) Initialize the parameters: Set the size of population as N. Set the evolution of generation as G. Set crossover probability as Pc. Set mutation probability as Pm.

(2) Set initial value of G as 0 and randomly generate the initial population as P0 whose number is N, then P0 = (x1, x2, ..., xn).

(3) Employ the third method to calculate fitness value of each organism of the present population. If the current generation is not the initial one, the fitness value is compared with mean fitness value of the past generation. If the fitness value is lower than previous one, the individual will be replaced by another good organism.

(4) Selection operator chooses two members of the present generation to participate in the next operations: crossover and mutation.

(5) If G<100, then adopt insertion mutation operation. Else adopt shift mutation operation.

(6) Determine whether the convergence criterion is satisfied, otherwise return to step 3.

IV. SIMULATION ANALYSIS

In order to testify whether the improved algorithm is effective, a number of experiments were done with different number of cities. The results show the significant differences between original and improved algorithm in the number of iterations, times, the computer time, and the calculated optimum length.

40 city coordinates were randomly selected. 703 generations were calculated applying the original algorithm and the best path long about 1997.67 was found (see Fig 3).

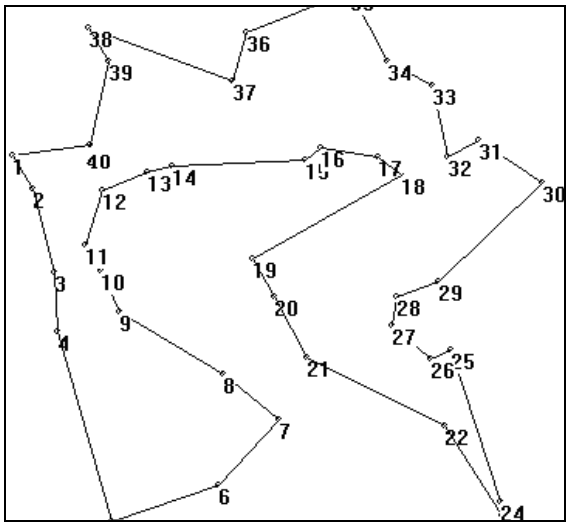


Figure 3. The best path found by the original algorithm.

The improved algorithm was employed to calculate the best path after 900 generations and found the shortest path about 1976.44 (see Fig 4). Same parameters were set in both the algorithms. The size of the initial population is 200; the crossover probability is 0.7; and the mutation probability is 0.2.

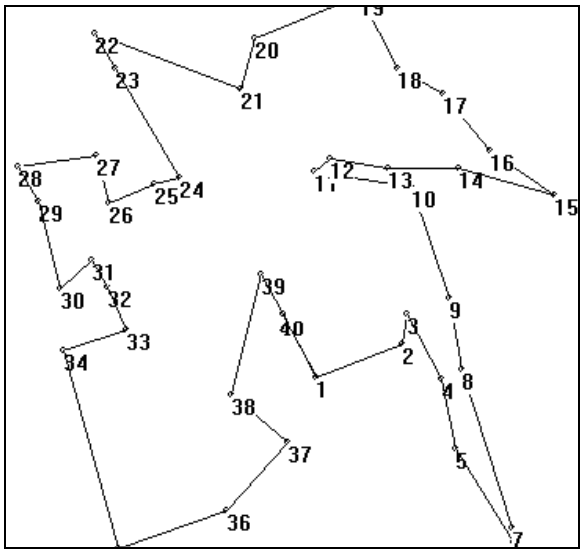


Figure 4. The best path found by the improved algorithm.

The improved algorithm works well if the maximum number of cities is smaller than 500. Fig 5 shows the best route after 5000 generations by the improved algorithm. But its calculation time is about 3 minutes. If the maximum number of cities is more than 1000, iteration time will be much longer. From the results, we can see that the effect of the improved algorithm is much better than that of original algorithm if the maximum number of cities is no more than 500.

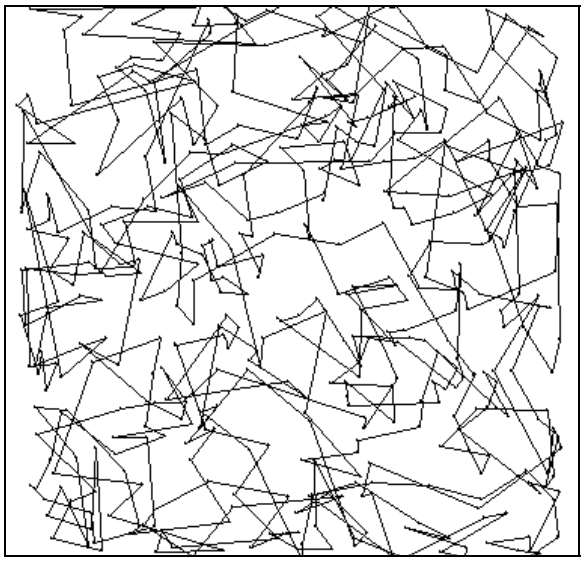


Figure 5. The best path found by improved algorithm with 500 cities.

The results are shown in the table II if numbers of cities are 40 and 100, respectively. In the table SAG means the original algorithm and IGA means the improved algorithm. There is no significant difference when number of cities is 40. However, when numbers of cities are larger, the more satisfactory results, not only in time but also in the length, can be seen in the improved algorithm.

Table II. Comparison between two algorithms.

Cities number	Generation		Shortest Rout Length		Time	
	SGA	IGA	SGA	IGA	SGA	IGA
40	762	649	2353.86	2280.45	3	3.5
	514	520	2723	2476.18	3	4
	388	353	2517	2336.74	3	3.5
100	1016	1026	6060.56	5298.57	20	20
	1026	978	5653.14	5543.98	20	17
	795	985	6204.64	5375.73	15	19

V. CONCLUSION

After nearly a century, great progress of research on methods for solving the TSP has been made. However, the problem has not been completely solved. TSP is still a hot issue at present. The original genetic algorithm is prone to early maturity or too fast convergence, which always lead to partially solving the TSP. In this paper, improving organism fitness calculation and choosing different strategy according to the different stages have been employed for TSP solving. Experimental results show that with the increase of the numbers of cities the improved algorithm technique can result in more satisfactory results in both computing time and in the best route finding compared with the original algorithm.

REFERENCES

[1] Holland J H. Adaptation in Nature and Artificial Systems. Ann Arbor, University of Michigan Press, 1975.

[2] Davis L D, Handbook of Genetic Algorithms, Van Nostrand reinhold, 1991.

[3] Junger M, Reinelt G, Rinaldi G. Handbooks in OR&MS: Chapter 4 the Traveling Salesman Problem. Elsevier Science, 1995.

[4] Merz P, Freisleben B. Genetic algorithms for the traveling salesman problem. Complex System, 2001.

[5] Greenhalgh D, Marshall S. Convergence Criteria for Genetic Algorithms. SLAM Journal of Computing, 2000, 30(1):269-282.

[6] Gibbs M S, Maier H R, Dandy G C, et al. Minimum Number of Generations Required for Convergence of Genetic Algorithms. Proc of IEEE Congress on Evolutionary Computation. Vancouver, Canada, 2006, 565-572.

[7] Rigal L, Tuffet L. A New Genetic Algorithms Specifically Based on Mutation and Selection. Advances in Applied Probability, 2007, 39(1): 150-161.

[8] L. M. Gambardella, M. Dorigo, Ant-Q: a reinforcement learning approach to the traveling salesman problem. Conf. On Machine Learning. Morgan Kaufmann, 1995, pp.255-260.

[9] J. A. K. Suykens , M. E. Yalcin and J. Vandewalle "Coupled chaotic simulated annealing processes", IEEE ISCAS, pp.582 -585 2003

[10] Zhou P, Li X P, and Zhang H F, "An ant colony algorithm for job shop scheduling problem", Proceedings of the 5th World Congress on Intelligent Control and Automation, Hangzhou, P.R. China, pp. 2899-2903, 2004.

[11] SHI X. H, XING X. L. and WANG Q. X. et al. "A Discrete PSO Method for Generalized TSP Problem," The 3rd International Conference on Machine Learning and Cybernetics. Piscataway, NJ: IEEE Press. 2004, pp. 2378-2383.

[12] Yang, Ning ,Li, Ping ,Mei, Baisha, An Angle-based Crossover Tabu Search for the Traveling Salesman Problem, Natural Computation, 2007. ICNC 2007. Third International Conference. Volume: 4, p: 512-516.

[13] R.N. Ma, Y.M. Xi, et al, "Dynamic Behavior of Discrete Hopfield Neural Networks with Time-Delay," Chinese Journal of Electronics, Vol.14, pp.187-191, 2005

[14] R.N. Ma, G.Q. Bai, "Stability Conditions for Discrete Hopfield Neural Networks with Delay," Lecture Notes in Computer Science, Vol.4113, pp.476-481, 2006.

[15] H. Nozawa, "A Neural-Network Model as a Globally Coupled Map and Applications Based on Chaos," Chaos, vol. 2, no. 3, pp. 377-386, 1992.

[16] L.N. Chen and K. Aihara, "Chaotic Simulated Annealing by a Neural Network Model with Transient Chaos," Neural Networks, vol. 8, no. 6, pp. 915-930, 1995.

[17] Zhang Rong, and Peng Hong, "A faster simulated annealing algorithm for the data clustering and its application", Computer Engineering and Applications, Vol. 15, No. 1, pp. 85-87, 2001.

[18] Dorigo M, Maniezzo V, and Colorni A, "Ant system: optimization by a colony of coorperating agents", IEEE Transactions on SMC, Vol. 26, No. 1, pp.29-41, 1996.

[19] Raman, Dhamodharan, Nagalingam, Sev V.; Gurd, Bruce W. A genetic algorithm and queuing theory based methodology for facilities layout problem. International Journal of Production Research, 2009, 47(20):5611-5635.

[20] DONG Yong-feng, GU Jun-hua, LI Na-na, HOU Xiang-dan, YAN Weili, "Combination of genetic algorithm and ant colony algorithm for distribution network planning," Machine learning and cybernetics, pp. 999-1002, Aug 2007.

[21] Tang Mi-nan, Enen Ren, ZHAO Chun-yan, "Route optimization for bus dispatching based on genetic algorithm-ant colony algorithm," Information Management, Innovation Management and Industrial Engineering, vol.4, pp.18-21, Dec 2009.