A Study of Solving Traveling Salesman Problem with Genetic Algorithm

Chutian Sun
Department of Industrial Engineering, Huazhong University of Science and Technology,
Wuhan 430000, China
e-mail: 1409655364@qq.com

***Abstracts*—Travelling salesman problem is one of the most important problems in the optimization area. To solve the problem of traveling salesman, the GA (genetic algorithm) can be seen as an appropriate method. In the genetic algorithm, there are many parameters needing to be set in advance. To figure out the effects of parameters in GA, we conducted experiments with different parameters and compared the performance. In addition, we try to improve the algorithm in aspects of crossing and mutating. The improved GA obtained a better performance, showing the proposed improvement is efficient.**

Keywords-the traveling salesman problem; the genetic algorithm; sensitive analysis; improved genetic algorithm

I. INTRODUCTION

The Traveling Salesman Problem (TSP) is a well-known problem of route selection and scheduling problems. It is to find the shortest route linking the set of cities based on the given distance between them. The arching route is required to go through all cities, but one city once, and finally the man must return to the origin one. It is also known as a classical NP-hard combinatorial problem in computer science and operations research.

The Traveling Salesman Problem was recorded as early as 1759 with the content of having a knight visit each of the 64 squares of a chessboard precisely once on its tour. In 1932, a practiced travelling salesman firstly mentioned the word of 'traveling salesman' in a German book. Hereafter, TSP was proposed by the Rand Company in 1948 and due to the Corporation's reputation, it became well-known and popular [1]. In addition, the new subject of linear programming and attempt to solve combinatorial problems was another reason why The TSP became popular at that time.

As a typical issue, TSP can be applied in plenty of practical scenarios, such as printing the circuit board, reforming the gas turbine engine，the science of X-Ray crystallography, etc [2].

In order to deal with the problem of TSP, many research raised a lot of optimizing algorithms. One of the earliest formulations was posed by Dantzig, Fulkerson and Johnson (1954), in which the objective function clearly described the cost of the optimal tour, though with the adding of cities, the formulation could not be directly solved by means of an LIP code [3]. In 1960, Miller, Tucker and Zemlin proposed an alternative formulation based on the Dantzig's formulation to reduce the number of subtour elimination constraints [4]. However, it was proved that Miller's method has a weaker linear relaxation than Dantzig's, though its relative compactness is effective [5].

In recent years, people found that many nature processes that are approaching steady state could be regarded as processes of optimization. Inspired by it, people proposed kinds of algorithms which have good performance in solving TSP. In 1989, Malek, Guruswamy and Pandya implemented parallel simulated annealing, and proved it is more effective than traditional annealing algorithm [6]. In 2013, two new neighborhood mechanisms were used to improve the searching performance of the simulated annealing algorithm in solving TSP [7]. If we ompared the improved algorithm with the traditional annealing algorithm, we can find that the improved algorithm takes on better solution and takes shorter time. In 1997, Dorigo and Gambardella adopted the ant colony algorithm to solve the problem of TSP [8]. The experiment proved that the algorithm can provide a good solution for both symmetric TSP and asymmetric TSP. In 2015, Mahi, Baykan and Kodaz employed the optimization of particle swarm and the 3-OPT algorithms to enhance the performance of the ant colony algorithm, which increases the performance of the solution and robustness [9].

Although there are many other algorithms to solve the problem of TSP, in this paper, we decide to use the GA to figure out the problem. In the early 1970s, John Holland came up of the concept of "genetic algorithm", which can be explained and is also inspired by evolution in biology such as genetic inheritance and genetic mutation, survival selecting and genetic crossing, providing the program with a technique of automatically improving their parameters [10]. On the basis of natural selecting and natural genetics, GA is a kind of searching algorithm, which is seen to be self-adapting, heuristic, global and probabilistic. This algorithm is designed to compute to find feasible and approximate solutions to the optimizing problem.

When the genetic algorithm is to be starting, we need to set a set of initial solutions. In the algorithm, solutions are also called populations. A solution is shown as a chromosome. The size of population for each generation is constant. To obtain the next generation of chromosomes with better performance, the fitness of each chromosome is evaluated and then selected probabilistically according to the fitness value. Some of the selected chromosomes randomly mate and produce offspring with random crossovers and mutations. In the race for survival of biology, a higher fitness value means the individual has more chance to be surviving.

978-1-7281-4306-4/20/$31.00 ©2020 IEEE

Correspondingly, we can say it will be chosen with a higher probability. So chromosomes of the new generation may have higher average fitness value than those of the old generation. Before meeting the end condition, GA will repeat the evolutionary process.

In 1985, aiming at the problem of TSP, Grefenstette firstly made use of the genetic algorithm to solve it. In recent years, many people have improved the Genetic algorithm and got a better performance in solving TSP [11]. Deng, Liu, and Zhou (2015) developed a new initial population strategy to improve the Genetic algorithm [12]. They firstly used k-means algorithm to group a set of cities. Then they the local optimum of each group can be found by using GA and we can acquire the global optimum of all groups. Finally, in line with the global optimal result, they disconnected the edge of local optimal path and reconnected the front and back groups (sort by cluster center distance). Bennaceur and Alanzi (2017) proposed a new crossover operator ESCX which is efficient in searching [13]. Compared to SCX, ESCX evaluates the cost of remaining tour and selects the next node based on this evaluation to build the offspring. The evaluation method adopts a greedy algorithm. To avoid the time consuming, the cost of remaining tour is estimated as the minimum cost of leaving the node.

In the process of GA, it can be found that the probability of mutating and crossing are significant. In this paper, we will study how these parameters influence the algorithm. In addition, we will try to improve the simple Genetic algorithm to speed up convergence.

II. METHOD

This study examines the classic TSP problem, where travelling salesman starts from the starting point and pass through all cities once. In the example, the city numbers and the parameters of mutating probability and crossing probability of GA will be changed, and we will observe how the optimal solutions and convergence speed change.

III. PROBLEM DESCRIPTION

In this paper, the city's number is unsure, and we can change the number to observe how the convergence speed changes. The location of cities is relative between each other, we will generate some random numbers as the distances between two cites, which is shown as follows:

TABLE I. TABLE CITY DISTANCE

City No. / Distance	1	2	3	4	5	6	7	8	9	10
1	0	70	65	56	76	35	45	87	49	57
2	70	0	67	56	48	67	91	24	37	38
3	65	67	0	48	47	41	30	84	46	28
4	56	56	48	0	41	29	37	36	24	57
5	76	48	47	41	0	35	53	51	27	38
6	35	67	41	29	35	0	27	92	38	56
7	45	91	30	37	53	27	0	47	56	78
8	87	24	84	36	51	92	47	0	34	36
9	49	37	46	24	27	38	56	34	0	45
10	57	38	28	57	38	56	78	36	45	0

A. The Simple Genetic Algorithm

To solve the problem of TSP, GA specifically includes six running steps: initializing, evaluating individuals, conducting to select optimal individuals, crossing genes and mutating, and a new population will come into being. The process of running GA is following:

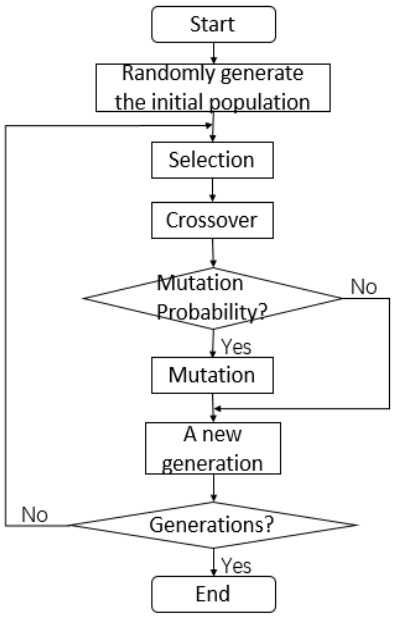

Figure 1. Operation Process

Before running the algorithm, we need firstly to decide how to implement coding. In this study, we use decimal coding to encode the sequence number of the city. A full permutation of sequence numbers is considered as a chromosome, which is a term of settlement to the problem. Here, the city's number is denoted as m.

Initialization is to produce an initial solution. We set the initial population size as N, and then N populations of gene with length m are randomly generated. Based on the population, the genetic algorithm continuously runs and iterates.

To select the individuals having better performance, we need to determine a fitness function. TSP requires the shortest path, so we choose reciprocal of the path length as the fitness function. That is, the fitness function in the algorithm is f(x) = 1/(sum of Distance). Based on the roulette strategy, individuals with greater fitness have a greater probability of being selected.

For updating the individuals, we conduct two steps: crossover and mutation. Based on the crossover probability, select the individuals to be crossed and pair them as parents. Two crossover sites are randomly selected on the chromosomes, and the genes between crossover sites are crossed. After the crossover, check whether the same code appears on the chromosomes, and if it does, the code does not participate in crossing. In view of the mutating probability, some genes will be selected to be mutating. Two mutation sites are randomly selected on the chosen

chromosome and genes at these position are reversed. After above steps, new population is generated and a new iteration continues.

The parameters need to be chosen carefully to solve the problem efficiently. Larger population size means more sample size and prevents premature convergence. However, larger population size increases the effort to calculate individuals' fitness, which causes slower convergence. By the operation of crossover, the new generation obtains a different structure from their parents. When it is too high for crossing probability, too much genetic will be changed and it is more likely that the excellent structure that forms before is lost. However, too low probability to cross would cause stagnation of the search. In its most basic form, we set the probability of crossing to be about 0.6 to 1.0. For the mutation probability, due to mutating operation bringing diversity to the population, too low mutation probability makes the search narrow and prevents information lost before recovering, while too high mutation probability causes the search becoming random.

B. Disadvantages of Genetic Algorithm

Before analyzing the parameters and designing the improved genetic algorithm, it is necessary to explore the advantages and disadvantages of the simple genetic algorithm. First, the following aspects show the shortcomings of GA: no need using additional information for the algorithm, low possibility of local optimization in the retrieval process, extensive expression of feasible solutions, extensibility, and the convenience of combining with other technologies. In turn, we expound the disadvantages of GA, such as lack of effective quantitative analysis, low efficiency compared with other optimization methods, difficulty in systematic expression of relevant constraints of optimizing, and irregularity of coding, etc.

C. Improved Genetic Algorithm

To improve the efficiency of genetic algorithm, two strategies are proposed. First, to make the mutation more directly, the mutation operation is improved as follow: for each city which is going to be mutated, the next position it is going to depends on the distance between the city and other cities. The shorter the distance, the greater probability of the mutating site going to that city. Second, to speed up the convergence, we tried to delete the paths with crossing edges. Suppose a path {a, a+1} is crossed with another path {b, b+1}, then we reverse the cities between a and b (excluding a).

D. Computer Simulation Analysis

In the following analysis, we use the software of MATLAB R2018a on a computer with Intel Core i7-6700HQ 2.60 GHz processor to simulate the problem of TSP, and analyzing the results comparing with those of the simple genetic algorithm.

E. Parameter Settings

In simulation, we set the population size $N = 100$, the max generations $t_{max} = 100$, the possibility of crossover $P_c = 0.9$, the possibility of mutation $P_m = 0.1$ and the number of cities $\mathrm{m} = 50$. For the distances of cities, we produce a group of random numbers.

F. Experimental Results

In the study, we will present two sets of experiments on both the classic genetic algorithm and improved genetic algorithm. The first experiment is using the stochastic cities to conduct sensitivity analysis, comparing its solution and running time when changing parameters of crossover probability and mutation probability. The second experiment is selecting a group of fixed cities to compare its running time and final result between the classic and improved algorithm.

a. Sensitivity analysis

In this study, we firstly analyze the effect of parameters. By setting different parameters, we can get the results in Table II. Each solution and iteration is average of 10 runs.

TABLE II. TABLE PARAMETERS CHANGING

P-Crossover	P-Mutation	Solution	Iteration
0.6	0.15	592.7162	478
0.7	0.15	580.3569	482.5
0.8	0.15	620.3467	456.6
0.9	0.15	611.6876	477.4
1	0.15	586.1623	470.7
0.8	0.05	708.4519	474.5
0.8	0.1	626.6199	477.3
0.8	0.15	620.3467	456.6
0.8	0.2	581.7184	470.3

In the following chart, we change the parameters of crossover probability and mutation probability. To express the results better, we draw some histograms:

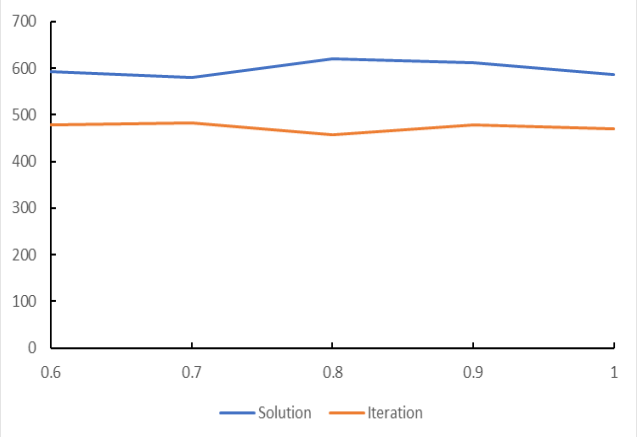

Figure 2. Parameter Changing of Crossover Probability(Mutation Probability =0.15)

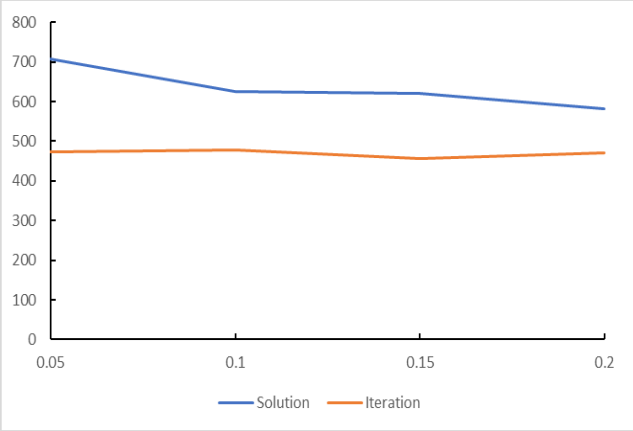

Figure 3. Parameter Changing of Mutation Probability

From the above figures, we can find that the parameters of crossing and mutating probability can affect the solution and iterations. The result shows that with the pre-determined city numbers and mutation probability, the solution become worse first and become better latter but the number of iteration has the opposite result when the crossover probability increase from 0.6-1. The Figure 2 shows that a best crossover probability may be 0.7. However, an increasing mutation probability has little effect on the number of iteration and get better solution. Comparing the results, we refer that mutation probability has a larger effect on the performance of GA.

b. Result comparison

In this study, we select a group of cities with fixed coordinates, and the cities are as follows:

TABLE III. CITIES' COORDINATES

City No.	X	Y
1	60	200
2	180	200
3	80	180
4	140	180
5	20	160
6	100	160
7	200	160
8	140	140
9	40	120
10	100	120
11	180	100
12	60	80
13	120	80
14	180	60
15	20	40
16	100	40
17	200	40
18	20	20
19	60	20
20	160	20

As shown in the above chart, we select 20 cities, whose serial number is from 1 to 20. Then we can use the classic genetic algorithm to obtain the following result:

Figure 4. Time of Running

When the iteration is 100, the running time is 14.151220 seconds.

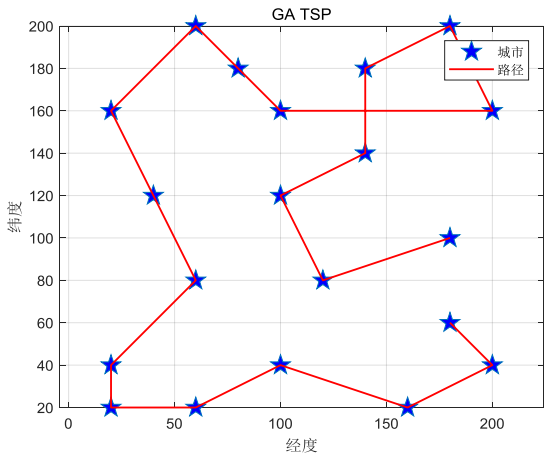

Figure 5. Path

So using the classic genetic algorithm, we can get the value of shortest path is 850.22 km. And the shortest path can be represented as 11-13-10-8-7-2-4-6-3-1-5-9-12-15-18-19-16-20-17-14.

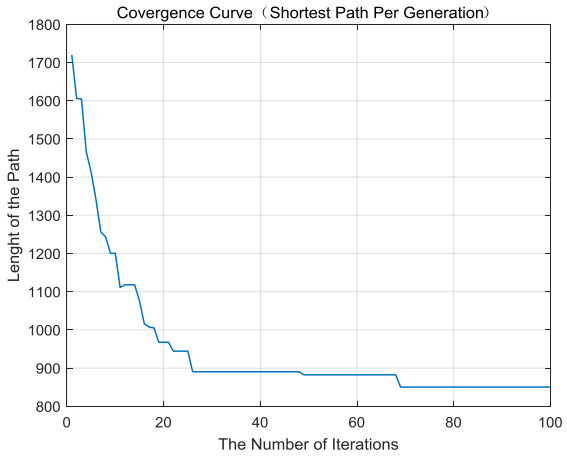

Figure 6. Convergence Curve

The above figure shows the convergence curve, representing the shortest path per generation. We can see that the classic genetic algorithm has a good convergence effect. And when the iteration is about 24, it can reach a better result.

For the improved genetic algorithm, we can get the following results:

```
Gen:98   The shortest path:908.59KM
    2    7    4    8   11   13   16   14   17   20   19   18   15   12    9    5    1    3    6   10

Gen:99   The shortest path:898.24KM
    2    7    4    8   11   13   16   14   17   20   19   18   15   12    9    5    1    3    6   10

Gen:100   The shortest path:898.24KM
    7    2    4    8   11   13   16   14   17   20   19   18   15   12    9    5    1    3    6   10

Time:0 minutes  12.994666 seconds.
```

Figure 7. Time of Running for the Improved Genetic Algorithm

When the iteration is 100, the running time is 12.99466 seconds, less than the time when using the classic genetic algorithm.

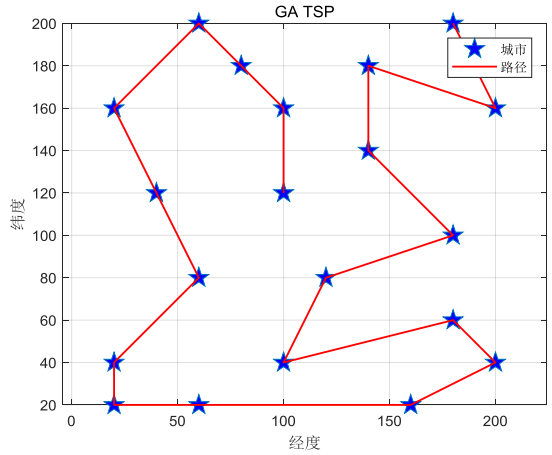

Figure 8. Path for the Improved Genetic Algorithm

So using the classic genetic algorithm, we can get the value of shortest path is 898.24 km. And the shortest path can be represented as 7-2-4-8-11-13-16-14-17-20-19-18-15-12-9-5-13-6-10.

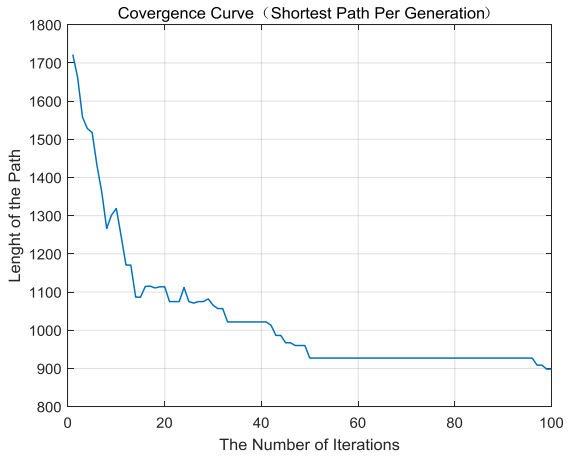

Figure 9. Convergence for the Improved Genetic Algorithm

The above figure shows the convergence curve of the improved genetic algorithm. And when the iteration is about 50, it can reach a better result.

From the above results, we can see that the improved algorithm needs less running time and can increase convergence rate, but the final result does not improve a lot.

IV. Conclusion

This paper has two major contributions. First, we solve the traveling salesman problem using the genetic algorithm, and set some different parameters to analyze its changing effects. The result shows that the mutation probability and crossover probability affect GA's performance, and the mutation probability has a more significant influence than crossover probability. Based on the simple genetic algorithm, we improve the algorithm with modifying the mutation step and deleting crossing paths. The improved algorithm is found to have a better performance, both in the running speed and the solution. In the future, more about the problem of traveling salesman and the genetic algorithm can be investigated.

References

[1] Bennaceur, H., & Alanzi, E. (2017). Genetic algorithm for the travelling salesman problem using enhanced sequential constructive crossover operator. International Journal of Computer Science and Security (IJCSS): 11(3), 42.

[2] Laporte, g. (1992) The Traveling Salesman Problem: An overview of exact and approximate algorithms. European Journal of Operational Research: 59 (1992), 231-247

[3] Dantzig, G., & Johnson, R. F. (1954) Solution of a large-scale traveling-salesman problem. Journal of the Operations Research Society of America: 2(4), 393-410.

[4] Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. Journal of the ACM: 7(4), 326-329.

[5] Langevin, A., Soumis, F., & Desrosiers, J. (1990). Classification of travelling salesman problem formulations. Operations Research Letters: 9(2), 127-132.

[6] Malek, M., Guruswamy, M., Pandya, M., & Owens, H. (1989). Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Annals of Operations Research: 21(1), 59-84.

[7] Hüsamettin, B, & Ramazan, S. (2013). A new simulated annealing approach for travelling salesman problem. Mathematical and Computational Applications, 18(3), 313-322.

[8] Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on evolutionary computation: 1(1), 53-66.

[9] Mahi, M., Baykan, Ö. K., & Kodaz, H. (2015). A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. Applied Soft Computing: 30, 484-490.

[10] Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor, MI: The University of Michigan Press.

[11] Grefenstette, J., Gopal, R., Rosmaita, B., & Van Gucht, D. (1985, July). Genetic algorithms for the traveling salesman problem. In Proceedings of the first International Conference on Genetic Algorithms and their Applications (Vol. 160, No. 168, pp. 160-168). Lawrence Erlbaum.

[12] Deng, Y., Liu, Y., & Zhou, D. (2015). An improved genetic algorithm with initial population strategy for symmetric TSP. Mathematical Problems in Engineering: 2015.

[13] Bennaceur, H. & Alanzi, E. (2017). Genetic Algorithm For The Travelling Salesman Problem using Enhanced Sequential Constructive Crossover Operator. International Journal of Computer Science and Security (IJCSS), Volume (11) : Issue (3) : 2017.