Genetic Algorithm for Solving The Travelling Salesman Problem

An Improved Implementation in Javascript

Parano Yang (09379037 杨超予)
School of Information Science & Technology
Sun Yat-Sen University
Guangzhou, China
Paranoyang@gmail.com

***Abstract*—In this paper, we develop an algorithm for quickly obtaining an optimal solution to Travelling Salesman Problem (TSP) from a huge search space. This algorithm is based upon the Genetic Algorithm which is mainly optimized from some previous related works. We employ roulette wheel strategy, survival-of-the-fittest strategy and survival-of-the-global-fittest strategy for selection mechanism. Besides, we use a heuristic operator for crossover and an inversion operator for mutation. We implement this algorithm and apply it into a TSP with 1000 cities.**

Keywords: TSP; GeneticAlgorithm; Selection Operator; Crossover Operator; Mutation Operator

I. INTRODUCTION

Genetic Algorithm (GA) that is first advanced by J.H. Holland professor, in 1975, is a type of directed random search algorithm based on natural selection used in computing to find approximate solutions for optimization and search problems (Genetic Algorithms, 2009; Holland, 1975). Main aim of this paper is to propose our method, which integrate a lot of previous works in this field, for achieving better results in Travelling Salesman Problem (TSP) using the Genetic Algorithm.

II. BRIEF DESCRIPTION ABOUT TSP AND GA

A. Travelling Salesman Problem (TSP)

TSP was first proposed in 1800s by W.R. Hamilton and the British mathematician Thomas Kirkman [14].It can simply describe as: a traveling salesman is going to n cities to promote his goods. He starts from some city c_0 and has to visit all of the rest cities only one time each. Finally he returns back to the starting city c_0. The problem is that we need to find the shortest route which meets the above requirements [15]. The travelling salesman problem is NP-hard but has many real world applications, making it a very popular problem to solve.

In this paper, $R = (c_0, c_1, \cdots, c_{n-1})$ represents the route. If R meets the condition that the value of formula (1) is the minimum, R is the best route.

$$f(R) = \sum_{i=0} d(c_i,c_{i+1}) + d(c_{n-0},c_0) \quad (1)$$

Among which, c_i represents the No. i city, n is the number of cities. $d(c_i,c_j)$ represents the distance form city c_i to city c_j. Obviously, we have $d(c_i,c_j) = d(c_j,c_i)$.

B. Genetic Algorithms

The Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. This technique has been used successfully in a variety of different problems, including the travelling salesman problem.

III. AN IMPROVED METHOD DESIGNED FOR SOLVING TSP

The different forms of encoding, crossover and mutation operator, or selection strategy that we have seen so far can be combined to give various genetic algorithms that can used to solve TSP. All the crossover routines can only be used with a certain form of encoding as well as the mutation operator. We believe that methods use heuristic information always performs better in this problem according some previous works [16]. We illustrate the detail in the following.

A. Encoding

Encoding is often the most difficult aspect of solving problem using GAs. Consider the TSP talked above. Our representation is simply a sequence of distinct numbers which represents a corresponding city in the map. For example, in a TSP with 10cities, we give each city a unique number from 0 to 9, and a sequence *{0, 2, 3, 1, 5, 6, 4, 8, 9, 7}* represent a trip which start from city 0, then city2, city3 … city7 and then, back to where it begins, city0.

B. *Evaluation*

Giving a certain trip in the encoding format given above, to evaluate the cost of this trip we use the Euclidean Distance to measure the cost of each sub-trip from city to city.

C. *Initialization*

After confirming the encoding mechanism, we randomly generate the initial population with the number of the constant value POPULATION_SIZE.

As for performance, we calculate all the Euclidean Distance between each two cities in a two-dimensional array in the initialization step.

D. *Fitness Function and Fitness Value*

TSP aims at finding out the individual with the minimum value of the objective function. As a result, we can treat the reciprocal of the objective function as the fitness function. And as to the equation (1), we have the following expression:

$$f(i) = 1/f(Ri)$$

The longer the trip is, the smaller the fitness value is, and vice versa.

E. *Selection*

The major task of the selection operator is to choose the individuals with higher fitness value. These individuals will reserve and be used to generate a new generation of population through crossover and mutation.

In this paper, we adopt the traditional roulette wheel selection [17] and an elite strategy of best-individual to survive. We make sure that the individual with the highest fitness value will be preserved into the next generation, and then use roulette wheel to get the other individuals to retain.

F. *Crossover*

Crossover operator is to generate new offspring. It processes the genes of the paternal chromosomes which are randomly grouped into pairs from the previous generation of the population, and to produce two offspring. As for TSP, the most commonly used crossover operator contains PMC, OC, CC and etc. [18][11].

In this paper, we adopt a heuristic crossover operator which uses the heuristic information that a shortest path trends to connect a city to another city which is much closer to it. The Algorithm was like this[19],[11]: first of all, with a predefined probability p which stands for the rate of crossover, pick out a certain amount of individuals as the parental crossover chromosomes, and then randomly make the parental chromosomes in pairs prepared for the crossover operator. Parental chromosomes are respectively marked as x and y in each pair. Next, randomly definite a position we can marked with city c as the starting point of the child traveling sequence. And now, the following paragraph gives a detailed introduction on how to get the offspring represented by *childx* and *childy*.

Generate new *childx*: calculate d_x and d_y which respectively means the distance between city c and the next adjacent city c_x, c_y in route x and route y. Compare d_x with d_y. If $d_x \leqslant d_y$ is true, c_x will be added to *childx*, else c_y will join in. After that, remove the city selected just now both in route x and in route y, in the meanwhile, set this city as the new city c, and repeat the above steps until there is only one city left in the parental chromosomes. Add the last city into *childx*, and now we obtain the first offspring *childx*. We must realize that the route is a closed-loop, which means the starting point is also the ending point.

Almost in the same way, we can generate new *childy* , and the only difference is that the meanings of d_x and d_y have changed. Under this condition, $_x d$ means the distance between city c and the previous adjacent city $\cdot c_x$ in route x, not the next one, and so is d_y.

G. *Mutation*

Mutation operator is use to change some certain partial chromosome' genes in a very low probability, which is going to keep the algorithm approaching to the optimal solution.

In this paper, we adopt two different heuristic mutation operators.

- The inversion operator.
- The shift operator.

IV. THE PROGRAM OF THE ALGORITHM

Finally, I implement this algorithm using Javascript which can draw the path and cities in any browser that support HTML5 canvas. The whole project was hosted on Github : https://github.com/parano/GeneticAlgorithm-Solving-TSP. and the online demo : http://errno.me/GeneticAlgorithm-Solving-TSP/. Another way is to download the source code on Github and use your browser to open the "./index.html" file.

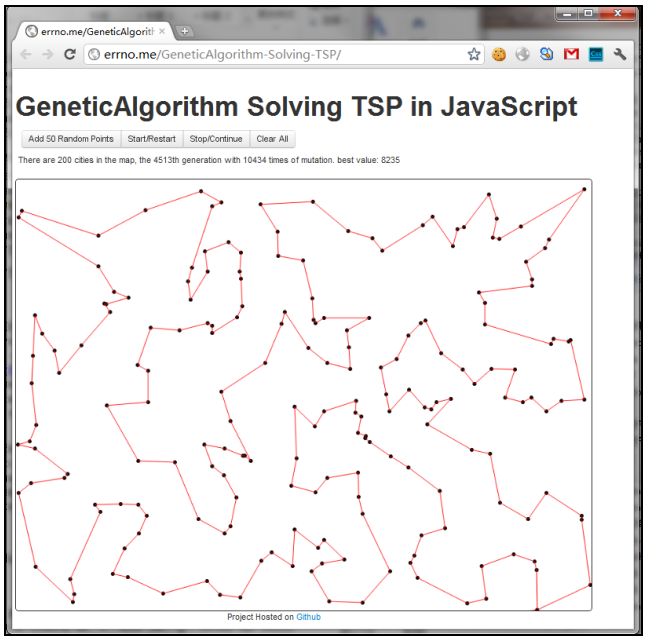

Figure 1. Default map with 200 cities.

By default, the map has 200 cities in it. Click the "Start" button in the web page, and you will see the evolution begins. Each time when this algorithm finds a better solution, it will immediately draw the newer one on the canvas so as to see how it looks like in the map as showed in Figure 1.

The canvas becomes totally blank after click "Clear All" button. Then you can add points to the map by click "Add 50 Random Points" which will randomly generate 50 cities in the map or by directly click on the map which will add a point to where you click. By clicking on the canvas, you can get any kind of TSP you want, like in the Figure 2.

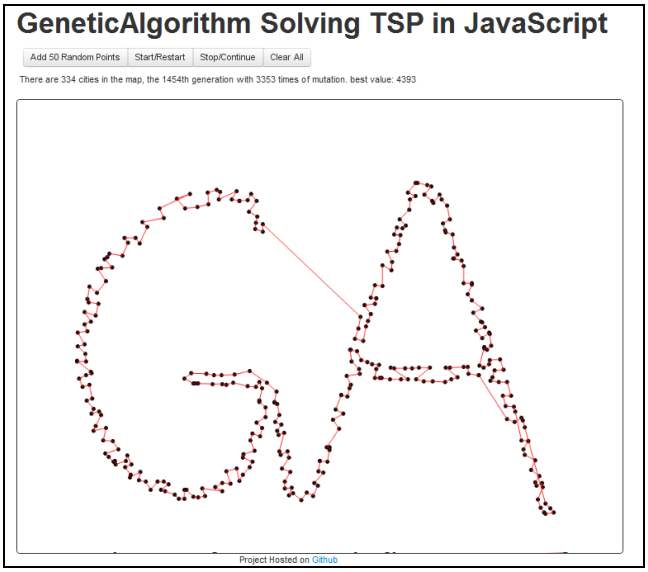

Figure 2. User specify map by clicking on the map.

In the program, we define some constant values as follow: the capacity of the population is 30, the probability of crossover is 0.9, and the mutation rate for both mutation operators is 0.005 which means the overall mutation rate is 0.01.

Figure 3. 500 cities after initialization

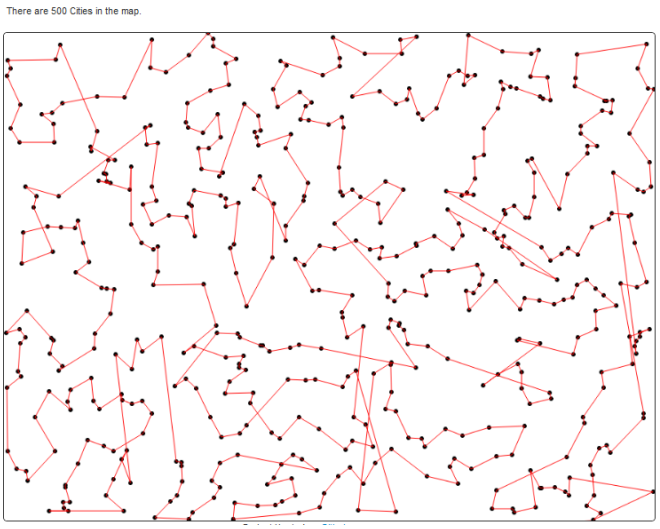

Figure 4. 500 cities, after about 100 generations

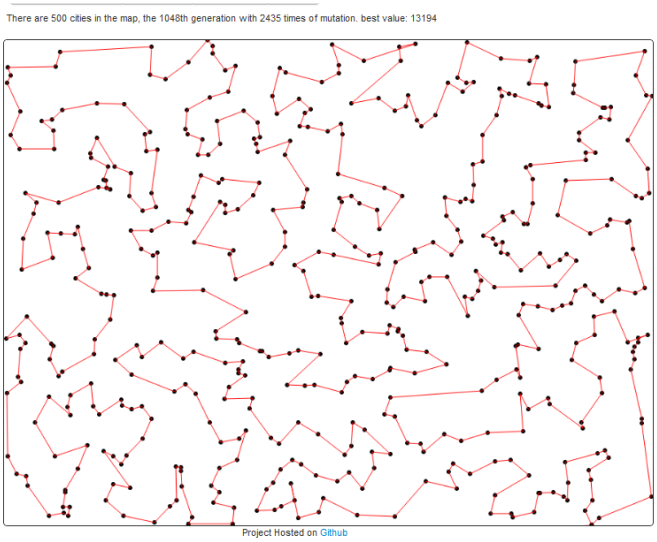

Figure 5. 500 cities, after about 1000 generations

In Figure 3 to 6, we give some screen shortcuts of the algorithm when solving a TSP with 500 random cities. We can see the initial route graph is becoming better and better during the evolution process. As in Figure4, a intermediate stage graph which has a lot of routes get overlapped which is obviously a very bad smell in solving TSP.

As in Figure5, most of the overlap routes has disappeared and it's hard to find a better solution manually. But until 3800th generation as showed in Figure 6, the optimal route is still changing and becoming better.

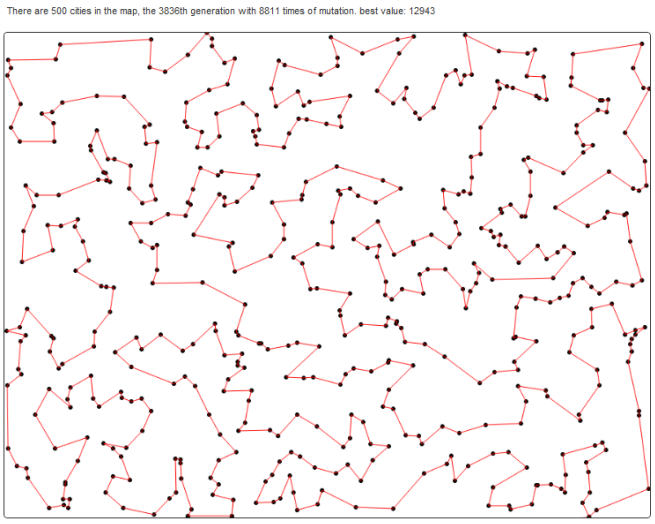

Figure 6. 500 cities, after about 3800 generations

We finally apply this algorithm to 1000 cities. Because of the whole program's implementation is using JavaScript and running under the browser process, the evolution process will become a little bit slow as to the restrictions of the browser. But after several thousand of generations, we can still get the optimal solution graph in Figure 7.

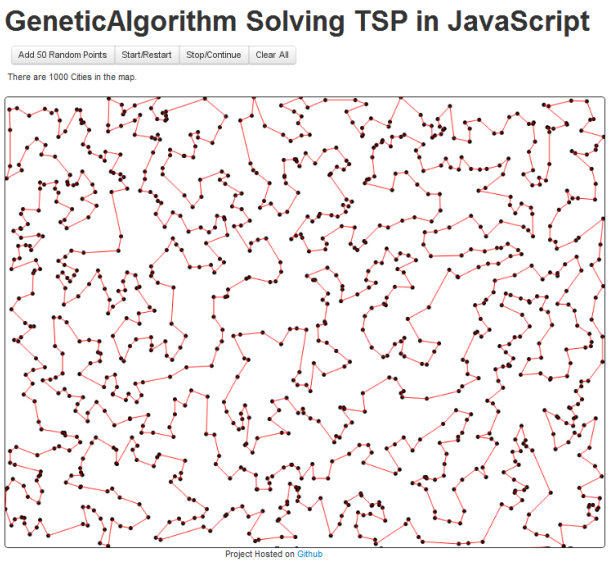

Figure 7. 1000 cities, after about 5000 generations

V. CONCLUSIONS

Genetic algorithms appear to find good solutions for TSP, however it depends very much on the choice of crossover and mutation operator. We believe that the methods that use heuristic information performs better in this area.

In this paper, we designed new algorithm structure which integrates three kinds of GA operators. All of these operators have their own heuristic information which can observably compare with other situations like changing the inversion mutation into a traditional swap mutation.

Overall, our algorithm can be used for finding optimal solutions to TSP with cities less than 1000. And drawing the best route in real-time also gives the program an intuitive way for considering the problem and algorithm.

REFERENCES

[1] Fox.B.R and McMahon.M.B .Genetic Operators for Sequencing Problems in[318].pp.284-300.

[2] Rawlms. G. Foundations of Genetic Algorithms, First Workshop on the Foundations of Genetic Algorithms and Classifier System, Morgan Kaufmann Publisher, San Mateo. CA. 1991.

[3] Gorges-Schlewer, M., ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy in [5].pp.422-427.

[4] Grefensteue, J.J., Gopal. R. Rosmaita B. and Van Gucht. D. Genetic Algorithm for the TSP in [5].pp.160-168.

[5] Grefensteue, J.J., (Editor).Proceedings of the First International Conference on Genetic Algorithms. Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[6] Grefensteue, J.J., Incorporating Problem Specific Knowledge into Genetic Algorithms.

[7] Davis, L.,(Editor), Genetic Algorithms and Simulated Annealing. Morgan Kaufmann Publishers, San Mateo. CA.1987.

[8] Jog.P.Suh, J.Y., Gucht, D.V., The Effects of Population Size, Heuristic Crossover and LocalImprovement on a Genetic Algorithm for the Traveling Salesman Problem.

[9] Liu Gang, Wang xuemei, Yang Lina, An Improved Genetic Algorithm and Its Application , ICNC 2010.

[10] Zhou Tao. TSP Problem solution based on improved Genetic Algorithm.

[11] Yingying Yu, Yan Chen, Taoying Li, A New Design of Genetic Algorithm for Solving TSP

[12] X. Bian, and L. Mi "Development on genetic algorithm theory and its applications," Application Research of Computers, vol.27, no, 7,pp. 2425-2429, 2010

[13] B. Chen, and H.Z. Xu, "An Improved Genetic Algorithm and Its Application in TSP," Computer Engineering, vol. 28, no.9, pp.90-92,. 2002

[14] D. Li, and H.X. Sun, "An Application Research of TSP Based on Genetic Algorithm," Science Technology of Heilongjiang Province, no. 13,pp.27,2009

[15] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), The Traveling Salesman Problem, ISBN 0-691-12993-2.

[16] Kylie Bryant, "Genetic Algorithms and the Traveling Salesman Problem", December 2000, Harvey Mudd.

[17] X.Y. Lai, "Genetic Algorithm and Its Application," China Computer and Communication, pp.117-119,2010.

[18] C. Fang, "Genetic Algorithm and Its Application in TSP Problem," China Computer and Communication, pp. 117-119, 2010.

[19] D.P. Peng, Z.Y. Lin, and J.Q. Wang, "An Improved Genetic Algorithm for TSP Problem," Computer Engineering and Applications, no.13, pp. 91-93,2006.

[20] 潘正君, 康立山, 陈毓屏,. 演化计算.