Solving Traveling Salesman Problems Using Ant Colony Optimization Algorithms

Zhiteng Cao, 16 Dec 2022

Abstract

Nature has always been the inspiration of human creativity, and the continuous evolution of organisms may inadvertently solve many complex problems. The ant colony algorithm is a heuristic algorithm based on nature. It simulates the process of ants looking for food. Through continuous iterations, ants will leave pheromones on the path, and pheromones will continue to decay during iterations. In this way, ants will continue to find the optimal path in iterations. The Ant Colony Algorithm has a good performance in many known NP-hard problems, but in this article, I will focus on the application of the ant colony algorithm to the traveling salesman problem. The traveling salesman problem is an NP-hard problem in combinatorial optimization, which involves a traveling salesman finding the shortest path in a series of cities so that the traveling salesman can visit all cities in the shortest time. On the TSP problem, compared with other heuristic algorithms, the advantage of the ant colony algorithm lies in its strong robustness and the ability to search for a better solution. But at the same time, it also has the problem of slow convergence speed and easy to falls into the local optimal solution. Therefore, the focus of this project will be to study how different parameters of the ant colony algorithm affect the convergence speed in TSP.

1. Introduction

1.1 Background on Ant Colony Optimization

Ant colony algorithm is a swarm intelligence algorithm, which is a group of non-intelligent or slightly intelligent individuals (Agents) that exhibit intelligent behavior through mutual cooperation, thus providing a new possibility for solving complex problems. Research [1] found that each ant leaves a substance called pheromone on the route it walks when foraging, and ants transmit information by sensing the concentration of this substance. When ants choose a path, they always tend to move towards the direction with high concentration of information cables, and ants walking on a short path leave more pheromones, and the probability of subsequent ants choosing it will be greater; The pheromones on other paths will continue to volatilize over time, thus forming a positive feedback mechanism, and finally, the entire ant colony gathers on the shortest path. Figure 1 below shows such a foraging process. [2]

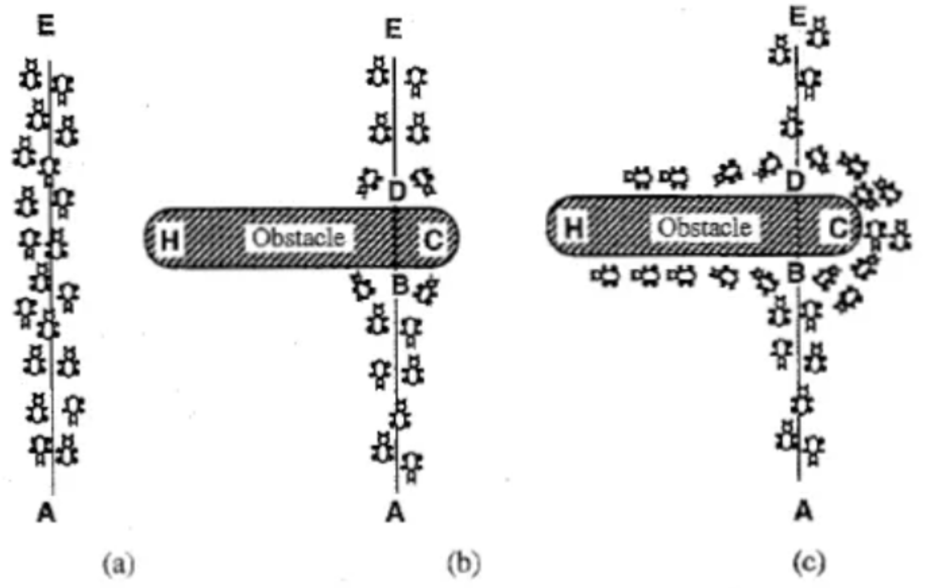


Figure (1)

In picture (a) of Figure 1, there is a group of ants, A is the ant nest, and E is the food. The colony of ants follows a straight path between the nest and the food. If an obstacle suddenly appears between A and E as shown in Figure (b), then the ants at point B will have to make a decision, they need to consider whether to move left or right. Since there is no pheromone left by the previous ants on the road at the beginning, the probability of the ants traveling in two directions is equal. But when an ant walks by, it will release a pheromone on its way, and this pheromone will be emitted at a certain rate. The ants behind it make decisions based on the concentration of pheromones on the road, whether to go left or right. As the left path is shorter than the right path, the pheromone will become more and more concentrated along the path on the right side as shown in figure (c), thus attracting more and more ants to travel along this path. [2]

1.2 Ant Colony Optimization on TSP

The description of the traveling salesman problem is: Knowing N cities and the distances between them, the traveling salesman starts from a certain city and traverses the N cities and the distances between them, then back to the starting city, determine a shortest path under the premise that each city is visited only once by the traveling salesman except the starting city. [4] The reason why I choose to use ant colony algorithm to solve the traveling salesman problem is that the ant colony algorithm has simple behavior rules, which make the algorithm have positive feedback and diversity. In the process of foraging, the diversity prevents the ants from going into a dead end and enters an infinite loop, while the positive feedback makes the ants constantly move towards the direction with high information concentration during the search process, so that the ants can find the shortest path or a path that is close to shortest path. The combination of the two feature makes the whole behavior intelligent which I think is quite interesting.

2 Related Work and Existing Solution

As a classic combinatorial optimization problem, the traveling salesman problem has been a research hotspot in the field of computer science in the past few decades. Nowadays, there are already many excellent algorithms that can solve the TSP problem, among which some optimization algorithms based on ant colony algorithm can achieve good results under different problem scales. This section will review a good work of modification of the ant colony algorithm.

In [2], the author proposed an ant colony algorithm based on the natural selection strategy. In this article, the author mentions that in the basic ant colony algorithm, when the ant colony performs the first iteration, the pheromone in each path induces the ants to be the same, and the factors for the ants to choose the path mainly depend on the distance between nodes. Such a design may cause the ants to choose a longer path as the optimal solution. In order to improve this problem, the author uses the quotient of city size and the distance between cities as the initial pheromone distribution matrix, thereby reducing the probability that the ant colony algorithm will fall into a local optimal solution due to the first selection of a shorter path. Therefore, the initial pheromone distribution matrix of the improved ant colony algorithm is shown in Figure 2.

$$t_{ij}(0)=\begin{cases} d_{ij}/n\ , & \textbf{if } i \neq j \\ 0\ , & \textbf{Otherwise} \end{cases}$$

Figure (2)

In addition, the author introduces two concepts of random evolution factor and evolution drift threshold according to the principle of survival of the fittest. Among them, the evolution drift threshold represents the variation threshold of the probability of the ant choosing the next path node, and the random evolution factor represents the probability compilation parameter of the ant choosing the next path node. The application of these two concepts is as follows; first, each ant obtains the probability of the next node according to the predetermined rules and then performs evolutionary selection on the probability obtained by each ant. [2] At this time, each ant will have a random evolution factor, and when the value of the random evolution factor is greater than the evolutionary drift threshold, the probability that the ant will choose the next node will vary, thus making the search path of the ant change. This design enables the ants to better adapt to changes in the environment during the search process, so that the ants can better find the optimal solution. The probability formula of the ants choosing the next node in the algorithm is shown in Figure 3.

$$P_{ij}^{k}(t)=\begin{cases}\dfrac{t_{ij}^{a}(t)\cdot ?_{ij}^{\beta}(t)}{\sum_{j\in allowed} t_{ij}^{a}(t)\cdot ?_{ij}^{\beta}(t)}, & \textbf{if } j\in allowed_{k}\\ 0 & , \textbf{otherwise}\end{cases}$$

$$P_{ij\text{-new}}^{k}(t)=\begin{cases}P_{ij}^{k}(t)\cdot rand, & \textbf{if } REF_{k}>EDT\\ P_{ij}^{k}(t) & , \textbf{otherwise}\end{cases}$$

Figure (3)

In order to verify the effectiveness of the improvement, the author respectively compared his improved algorithm, ant colony algorithm base on nature selection - ACANS, with the basic ant colony algorithm ACO, and a genetic ant colony hybrid algorithm based on the ant algorithm and genetic algorithm - H3AGA. [2] The results show that under the same personal computer environment, as the city size gradually increases, the convergence speed of ACANS is significantly faster than that of ACO and H3AGA, which proves the effectiveness of the improved algorithm. (Figure 4) [2]

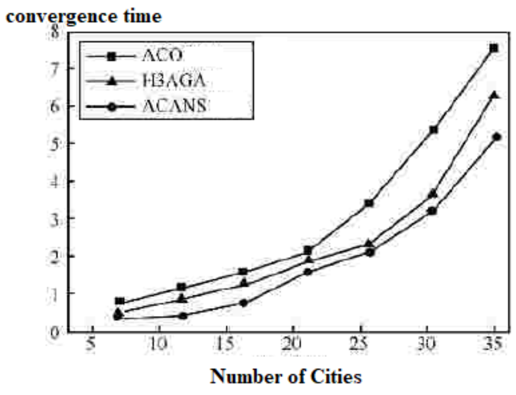


Figure (4)

3 Problem Approach

3.1 Basic Principle of ant colony algorithm on TSP

In the basic ant colony algorithm, the computer needs to set the number of ant populations according to the scale of the problem (e.g., map size, map geography), and let each ant start a parallel search. After every ant visits all the cities, it releases pheromones along the route, and the number of pheromones should be proportional to the quality of the solution. [5] If the ant's current city is i and the next city is j, the ant's choice of path will be determined according to the pheromone concentration Tij and the distance Dij between cities. A random local search strategy is used to make the pheromone concentration on the side with a shorter distance larger, so that the ants have a greater probability of choosing this side. Besides that, some restrictions include that each ant can only pass through each city once, which has a taboo table to control. [5] When all ants have completed a search, it is a complete iteration, and in each iteration, the pheromone on each edge will be updated. And after updating the pheromone, a new round of search will start. It is worth noting that the ants in the same iteration process will not be affected by the pheromone left by each other. Each pheromone update includes the evaporation of the last pheromone and the increase of pheromone on the path passed. When the algorithm reaches the predetermined number of iterations, it exits the algorithm and regards the current solution as the optimal solution.

3.1.1 Path Construction

At the beginning of the problem, each ant will be randomly generated to a city as its departure city, and at the same time, according to the number of ants (AntCount) and the number of cities (CityCount), a two-dimensional matrix is generated as a taboo table for storing each city where an ant has passed. The size of this two-dimensional matrix is AntCount*CityCount. When an ant chooses the next city j from city i, the transition probability pij is determined by two factors, one is the pheromone concentration Tij, and the other is the distance Dij between cities. Where alpha is the information importance factor, and beta is the heuristic function importance factor. At time t, the transition probability of ant k from city i to city j is calculated according to the following formula (Figure 5):

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\mathbf{T}_{ij}(t)]^{\alpha}[\mathbf{D}_{ij}]^{\beta}}{\sum_{k\in \text{ callowed }}[\mathbf{T}_{ik}(t)]^{\alpha}[\mathbf{D}_{ik}]^{\beta}}, & j \in \text{ allowed }_{k} \\ 0, & \text{else} \end{cases}$$

Figure (5)

In the formula of Figure 5, alpha and beta are adjustment factors, which are used to adjust the importance of Tij and Dij. In addition, allowed k is used to indicate the path that ant k has not traveled, and this part of information is represented by the taboo table mentioned above. [6] If the information concentration on the path i to j is greater, the value of Tij is greater, and the moving probability of ant k from city i to city j is greater. At the same time, if the distance between city i and city j is shorter, the value of Dij is smaller, and the transfer probability of ant k from city i to city j is greater.

3.1.2 Pheromone Update

In each iteration, the pheromone update includes pheromone evaporation and pheromone increase. The evaporation of pheromone means that the pheromone on each side will decrease with time, and rho is the evaporation factor of pheromone, which is used to express the evaporation speed of pheromone. [6] The increase of pheromone means that when each ant passes through an edge, it will leave a certain amount of pheromone on this edge. The amount of this pheromone is determined by the path length of the ant. The shorter the path, the more The more pheromones under it. In each iteration, the update formula of pheromone is (Figure 6) [6]

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\,\cdot$$

Figure (6)

In the formula in Figure 6, the first part on the right side of the formula represents the pheromone after volatilization, and the second part is the pheromone left by each ant on the path from city i to city j.

3.2 Approach to the Problem

As mentioned in the Abstract, although the ant colony algorithm has a good performance on the TSP problem, it has the problem of slow convergence and easy to fall into the local optimal solution. This article will make some adjustments to the parameters of the ant colony algorithm, so as to find the appropriate parameters to improve the convergence speed of the algorithm, so that the algorithm can find the global optimal solution faster. For comparison, this article will use att48 as the test data set, and compare and test the original ant colony algorithm and the improved ant colony algorithm respectively, so as to illustrate the effect of the improved ant colony algorithm. The values of alpha, beta and rho in the ant colony algorithm all have a great influence on the convergence speed of the ant colony algorithm. The role of alpha is to control the importance of pheromones. When the value of alpha is larger, the importance of pheromones is greater. That is to say, when ants choose the city to go to next, they will be more inclined to Choose a city with a high pheromone concentration. The role of beta is to control the importance of the distance between cities. When the value of beta is larger, the importance of the distance between cities is greater. That is to say, when ants choose the city to go to next, they will be more inclined to choose cities with short distances between cities. The role of rho is to control the volatilization speed of pheromones. When the value of rho is larger, the volatilization speed of pheromones is faster, that is to say, the volatilization speed of pheromones is faster, and the importance of pheromones smaller. This article will make some adjustments to the values of alpha, beta and rho to find parameters that can speed up the convergence speed of the ant colony algorithm when the city data set is att48.

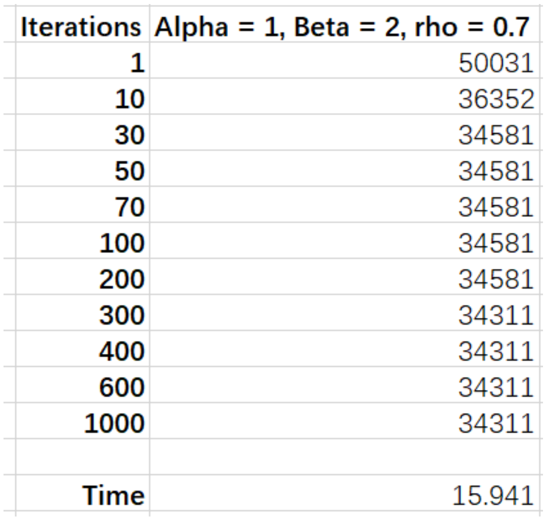
4 Experiment design and results

4.1 Experiment design

In order to compare the ant colony algorithm with default parameters with the ant colony algorithm with changed parameters, this paper will use the same att48 data set as the test data set. In the experiment, I will set the number of ants to 600, and set the value of epochs to 1000, so as to ensure that the algorithm can converge within a limited number of iterations. As for the parameters, the initial value of alpha is set to 1, the initial value of beta is set to 2, the initial value of rho is set to 0.7, and the initial values of these three parameters are set as the control group. As for testing, I will set the rho value to 0.3, 0.5, 1.0, and 1.5 in sequence, and for the alpha value, I will set it to 0.5, 0.7, 4, and 6 in sequence. For the value of beta, I will set it to 0.5, 1, 4, and 8 in sequence. For each set of different parameters, I will run it ten times and record the average solution and average running speed, so as to compare the impact of different parameters on the result generation of the ant colony algorithm. In order to ensure the repeatability of the experiment, I will use the same random seed to initialize the random number generator. By changing the value of a single variable compared to the control group, I will observe the effect of different parameters on the convergence speed of the ant colony algorithm.

4.2 Experiment results

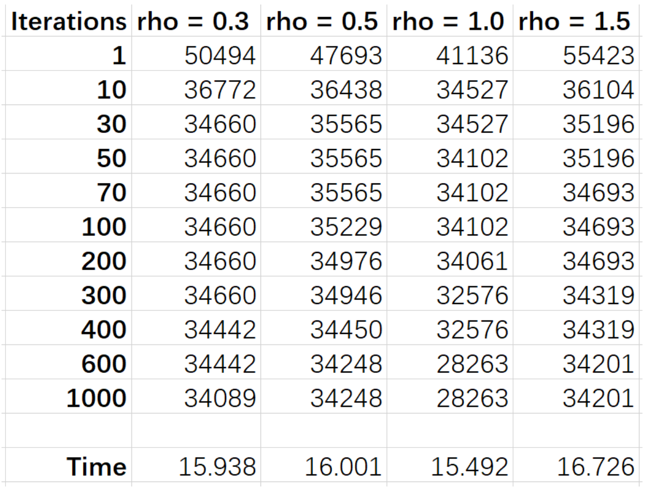
For the att48 data set, other than obtaining the control group data, I conducted 12 experiments by adjusting the three parameters of Alpha, Beta, and rho. My main concern with the experiments is to see how the various variables affect the rate of convergence and the speed of calculation. Figure 7 shows the running results of the Control Group. The left column shows different iteration times, while the right column shows the shortest path found by the ant colony algorithm in different iteration times when Alpha =1, Beta = 2, and rho = 0.7. In the bottom column, the time it takes to run 1000 iterations is recorded.



Iterations	Alpha = 1, Beta = 2, rho = 0.7
1	50031
10	36352
30	34581
50	34581
70	34581
100	34581
200	34581
300	34311
400	34311
600	34311
1000	34311
Time	15.941

Figure (7)

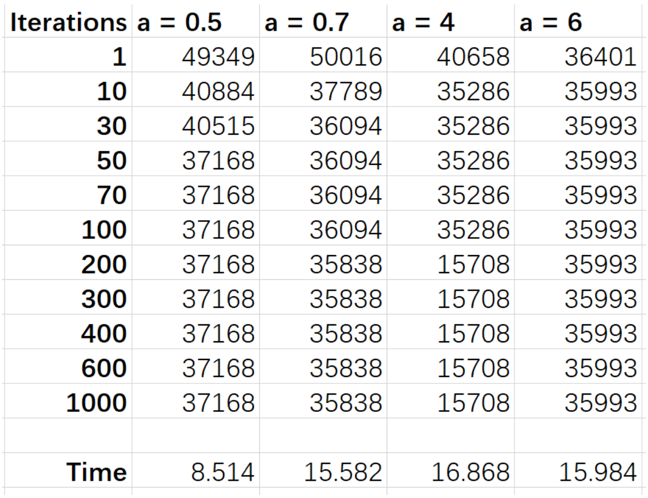
Similar to Figure 7, Figure 8 shows the shortest path found at different iterations when the rho value is different from the Control Group and the Alpha and Beta values are the same. From left to right are rho = 0.3, rho = 0.5, rho = 1.0, rho = 1.5. The bottom column shows the difference in computing time caused by different rho.



Iterations	rho = 0.3	rho = 0.5	rho = 1.0	rho = 1.5
1	50494	47693	41136	55423
10	36772	36438	34527	36104
30	34660	35565	34527	35196
50	34660	35565	34102	35196
70	34660	35565	34102	34693
100	34660	35229	34102	34693
200	34660	34976	34061	34693
300	34660	34946	32576	34319
400	34442	34450	32576	34319
600	34442	34248	28263	34201
1000	34089	34248	28263	34201
Time	15.938	16.001	15.492	16.726

Figure (8)

Figure 9 shows the shortest path found at different iterations when the alpha value is different from the Control Group and the rho and Beta value are the same. From left to right are Alpha= 0.5, Alpha= 0.7, Alpha= 4.0, and Alpha = 6.0. The bottom column shows the difference in computing time caused by different Alphas.



Iterations	a = 0.5	a = 0.7	a = 4	a = 6
1	49349	50016	40658	36401
10	40884	37789	35286	35993
30	40515	36094	35286	35993
50	37168	36094	35286	35993
70	37168	36094	35286	35993
100	37168	36094	35286	35993
200	37168	35838	15708	35993
300	37168	35838	15708	35993
400	37168	35838	15708	35993
600	37168	35838	15708	35993
1000	37168	35838	15708	35993
Time	8.514	15.582	16.868	15.984

Figure (9)

Figure 10 shows the shortest path found at different iterations when the Beta value is different from the Control Group and the rho and the Alpha value are the same. From left to right are Beta= 0.5, Beta= 1.0, Beta= 4.0, and Beta = 8.0. The bottom column shows the difference in computing time caused by different Betas.

Iterations	b = 0.5	b = 1	b = 4	b= 8
1	95032	69817	37232	34941
10	50652	37280	35636	34941
30	42607	37280	34108	34941
50	42081	36435	34108	34941
70	41919	36435	34108	34941
100	40628	36409	34108	34941
200	40628	35537	34108	34941
300	39966	35537	34108	34941
400	39966	35537	34108	34941
600	39966	35478	34108	34941
1000	38797	35208	34108	34941
Time	15.757	24.726	24.676	24.796

Figure (10)

5 Analysis of experiment result

Iterations	Control	rho = 0.3	rho = 0.5	rho = 1.0	rho = 1.5	a = 0.5	a = 0.7	a = 4	a = 6	b = 0.5	b = 1	b = 4	b= 8
1	50031	50494	47693	41136	55423	49349	50016	40658	36401	95032	69817	37232	34941
10	36352	36772	36438	34527	36104	40884	37789	35286	35993	50652	37280	35636	34941
30	34581	34660	35565	34527	35196	40515	36094	35286	35993	42607	37280	34108	34941
50	34581	34660	35565	34102	35196	37168	36094	35286	35993	42081	36435	34108	34941
70	34581	34660	35565	34102	34693	37168	36094	35286	35993	41919	36435	34108	34941
100	34581	34660	35229	34102	34693	37168	36094	35286	35993	40628	36409	34108	34941
200	34581	34660	34976	34061	34693	37168	35838	15708	35993	40628	35537	34108	34941
300	34311	34660	34946	32576	34319	37168	35838	15708	35993	39966	35537	34108	34941
400	34311	34442	34450	32576	34319	37168	35838	15708	35993	39966	35537	34108	34941
600	34311	34442	34248	28263	34201	37168	35838	15708	35993	39966	35478	34108	34941
1000	34311	34089	34248	28263	34201	37168	35838	15708	35993	38797	35208	34108	34941
Time	15.941	15.938	16.001	15.492	16.726	8.514	15.582	16.868	15.984	15.757	24.726	24.676	24.796

Figure (11)

Figure 11 is a collection of all test results. Through this table, we can intuitively find two extreme values. When a = 0.5, the average search time is the least, only 8.514 seconds. This is especially prominent in the result set where most test results are around 15 seconds. Another outstanding value is that when a=4, the shortest path found is 15708, which is also the shortest distance found in the entire data result set. The reason for such a result is that alpha is a pheromone importance factor, the larger its value is, the less likely the ant is to choose the path it traveled before, and the randomness of the search path is also weakened. This makes it easier for the algorithm to find a relatively better solution. And when the alpha value is smaller, the ant colony search range will be reduced, so it is easy to fall into a local optimal solution.

In the test of Beta value, we can see that when Beta is equal to 1, 4, or 8, the time spent in solving is similar, and the shortest distance obtained is also approximate. It is worth noting that when Beta is equal to 8, the solution is not updated from the first iteration. And when b=0.5, although the solution is not very superior, the time consumed is less. We can draw a conclusion that when the beta value is larger, the ant colony is more likely to choose a locally shorter path, and the convergence speed of the algorithm will be accelerated at this time, but the randomness is not high, and it is easier to obtain a local relative optimal solution.

In the test for rho, we can see that when the alpha and beta parameters are the same, different rho values are resulting having similar calculation time and calculation results. The reason why the test changes to the rho value did not bring excellent path search results is that when the rho value is too small, there are more pheromones left on each path, which will cause invalid paths to be searched continuously, thus affecting the convergence efficiency of the algorithm. When the rho value is too large, although the invalid path can be excluded from the search, the effective path may also be abandoned from the search, thus affecting the search for the optimal value.

6 Conclusion and Future Experimentation

From the test results, the setting of the Alpha value will have the greatest impact on the results of the entire ant colony algorithm. If a shorter running time is wanted, the Alpha value should be set smaller, and if a more accurate answer is wanted, the Alpha value should be set relatively larger. The Beta value should be set to too large. The selection of the value range of rho is more complicated, and needs to be set according to the actual situation of the problem and the configuration of other parameters. Through this project, I have a deeper understanding of the relationship between the Ant Colony Algorithm and its parameters. In future experiments, I plan to explore the relationship between Alpha, Beta, and Rho in more depth. Also, I would like to find out what would be a more suitable value range for each of the parameters under the different scales of problems.

References:

[1] Dorigo, Marco, Gianni Di Caro, and Luca M. Gambardella. "Ant algorithms for discrete optimization." Artificial life 5.2 (1999): 137-172.

[2] Xu K, Wu J, Huang T, Liang L. An Improvement of a Mapping Method Based on Ant Colony Algorithm Applied to Smart Cities. Applied Sciences. 2022; 12(22):11814.

[3] Wu,Huafeng and Chen, Xinqiang. "Improved ant colony algorithm based on natural selection strategy for solving TSP problem" Journal on Communications 34.3 (2013): 165-170.

[4] Reinelt, Gerhard. The traveling salesman: computational solutions for TSP applications. Vol. 840. Springer, 2003.

[5] Yang, Jinhui, et al. "An ant colony optimization method for generalized TSP problem." Progress in Natural Science 18.11 (2008): 1417-1422.

[6] Dorigo, Marco. “Ant Colony Optimization.” Scholarpedia, Scholarpedia, 28 Mar. 2007