Comparison of various algorithms based on TSP solving

JianChen Zhang*
College of Transport and Communications, Shanghai Maritime University, Shanghai, 201306, China.

* Corresponding author: 202030610111@stu.shmtu.edu.cn

Abstract. The traveling salesman problem (TSP) is a classic combinatorial optimization problem. As a hot research problem, it has been applied in many fields. Since there is currently no algorithm with good performance that can perfectly solve this problem, for the solution of the traveling salesman problem, this article first establishes a mathematical model of the traveling salesman problem, and uses four solving methods for the same case: the greedy method, Branch and bound method, dynamic programming method and genetic algorithm, analyze the applicability and accuracy of different algorithms in the same case.

Keywords: traveling salesman problem; greedy method; branch and bound method; dynamic programming method; genetic algorithm.

1. Introduction
TSP is not only an NP problem [1], it is also one of the most studied problems in combinatorial optimization. The so-called TSP problem can be simply described as: There is a salesman who wants to go to other cities to promote his own products. It is known that there are n cities, and the distance between them is known. Suppose the salesman starts from the starting city A., To go to other cities, each city must arrive once and cannot be repeated. Finally, you must return to the starting point city A. Therefore, we need to design a complete route to enable the salesperson to achieve the desired goal (for example, the least travel time, the shortest total length of the route, the least cost of travel, etc.)

2. Design case and build model
In order to facilitate the comparison and highlight the differences and characteristics of various algorithms, four algorithm greedy methods, branch-and-bound methods, and dynamic programming methods are used to compare with genetic algorithms. The city coordinate table and the city coordinate distribution map of the design case are shown below, where X and Y represent the horizontal and vertical coordinates of the city location, and the unit is km.

| City number | X    | Y    | City number | X    | Y    |
|-------------|------|------|-------------|------|------|
| 1           | 2066 | 2333 | 6           | 2253 | 478  |
| 2           | 935  | 1304 | 7           | 949  | 3025 |
| 3           | 1270 | 200  | 8           | 87   | 2483 |
| 4           | 1389 | 700  | 9           | 3094 | 1883 |
| 5           | 984  | 2810 | 10          | 2706 | 3130 |
To establish a TSP model, suppose a graph \( G = (V, E) \) is a weighted graph, where \( V = \{1, 2, 3, \ldots, n\} \) is the vertex set, \( E \) is the edge set, and suppose \( d_{ij} (i, j \in V, i \neq j) \) that is the distance between vertices \( i \) and \( j \). Among them \( d_{ij} > 0 \) and \( d_{ij} \neq \infty \), at the same time \( d_{ij} = d_{ji} \) \((i, j \in V)\). Then the mathematical model of the TSP problem is:

\[
\text{min } F = \sum_{i \neq j} d_{ij} x_{ij} \quad (1)
\]

\[
st, x_{ij} = \begin{cases} 1, & e_{ij} \text{ is on the optimal path} \\ 0, & e_{ij} \text{ is not on the optimal path} \end{cases} \quad (2)
\]

\[
\sum_{i \neq j} x_{ij} = 1, i \in V \quad (3)
\]

\[
\sum_{i \neq j} x_{ji} = 1, j \in V \quad (4)
\]

\[
\sum_{i \neq j} x_{ij} = |V|, V \text{ is a subgraph of } G \quad (5)
\]

Among them, \( x_{ij} \) represents the decision variable. When the value \( x_{ij} \) is 1, it means that the current edge \( e_{ij} \) is on the optimal path, and vice versa. Formula (1) is expressed as the objective function of the TSP problem, which is to find the minimum distance of the loop through all vertices. Since the optimal solution, that is, the shortest path, is unique, it must be ensured that there is only one shortest path for a given \((i, j)\). Therefore, formula (3) and formula (4) are needed to ensure that only one time can be achieved. Entering the shortest path means that the sum of only one side \( x_{ij} \) is 1, and the rest are 0. The absolute value in the formula (5) represents the number of elements in the set, and it limits that there will be no sub-loops in the loop.

3. Comparison of various algorithms to solve TSP problem

(1) Greedy Algorithm means that the decision made must have no aftereffect, that is, it does not affect the subsequent and previous strategies. In the process of solving the problem, only the optimal choice of the current situation is considered [3]. The key to the greedy method is how to choose a greedy strategy. It does not consider the overall problem, but only considers whether it is optimal locally. The solution idea of the greedy method is as follows:
The core idea of the algorithm is to select the current shortest path. Every time the next city is selected from the initial city, only the current optimal solution is considered, that is, only the shortest distance from the current city is considered every time when starting to the next city. After each selection, record the city that has been traversed, mark each city that has been visited, and calculate the total length of the minimum path at the same time, and finally calculate the total length of the path traversed once.

The final solution result is as follows:

The total distance traveled by the traveling salesman is about 10464.18km. The solution time is about 9.24e-5s.

(2) Branch and Bound is commonly used to solve integer programming problems. Its steps are mainly divided into three parts: branching, delimiting and pruning. Among them, branching refers to dividing all feasible solution spaces and continuously subdividing them into smaller subsets; delimiting refers to calculating an upper bound and a lower bound for each child node after searching for child nodes; pruning refers to after each branch, if the calculated lower bound is larger than the current optimal solution, this branch route is eliminated. This slowly narrows the scope of the search until a feasible solution is found [4]. The solution obtained by the branch and bound method is generally the optimal solution.

The solution idea of the branch and bound method is as follows:

In solving the TSP problem, the steps of the branch and bound method are as follows:

1) First, traverse the solution space tree according to the width-first strategy. The breadth-first strategy refers to searching all the child nodes every time it branches.

2) In the traversal process, delimit each node i, and estimate the value range of the objective function that may obtain a complete solution by searching down the node according to the limit function, that is, the limit $bound(i) = [down(i), up(i)]$
3) Select the node. The rule of selection is to select the node that can minimize the objective function from the limit obtained in the previous step. The search process also needs to be carried out according to the width-first strategy, and the search direction needs to be continuously adjusted to find a feasible solution as soon as possible.

In the process of delimiting, the greedy method is used to define the upper bound of the value range, and the average of the two minimum values in each branch of the space tree is taken as the lower bound of the value range. The formula is as follows:

\[
up^i(s) = \text{Greedy}(s)
\]

\[
down(i) = \sum \left\{ \frac{\min_1 + \min_2}{2} \right\}
\]

The final solution result is as follows:

![Fig. 5 Branch and Bound Method Solving Simple Case Results](image)

The total distance traveled by the traveling salesman is about 10127.55km. The time required for the solution is about 0.319s.

(3) Dynamic Programming is a classic algorithm strategy to solve optimization problems [5]. Its idea is to decompose the original problem \(X\) into several sub-problems (such as \(X_1, X_2, X_3...\)), and solve each sub-problem separately, and the current decision is affected by the previous decision, that is, the decision on \(X_1\) will affect the choices made at \(X_2\). Since the optimal solution of the original problem is derived from the optimal solution of the sub-problem, the solution table of the sub-problem needs to be stored in the calculation process to avoid repeated calculations. The solution idea of dynamic programming method is as follows:

![Fig. 6 Schematic diagram of dynamic programming method](image)

The basic idea of solving TSP problem is mainly divided into three parts:

1) Assume that the salesman travels through cities in accordance with the rules of the TSP problem, starting from the initial city, going through all the cities and returning to the origin at the end. The information that needs to be recorded is the current city location and the current untraversed city set \(s\). The binary form is used to represent \(s\), the traversed city is written as 1, and the untraversed city is written as 0.

2) Constructing the state transition equation, which is the core part of the dynamic programming method, which can continuously subdivide the original problem into various sub-problems. The definition of the state transition equation is as follows:

\[
T(s,\text{init}) = \min\{T(s-i,i) + \text{dist}[i][\text{init}]\}
\]
Among them, \( T(s,\text{init}) \) represents the total length of the path from the current point location to the initial point \( \text{init} \) after passing through the set \( v \) of experienced points, and \( s \) is an empty set at the end.

3) Finally, a \( T \) table needs to be constructed to record the value of \( T \). If each search has to be recalculated, it becomes a brute force search. Therefore, it is necessary to construct a \( T \) table \( dp[s][\text{init}] \) to record the value of \( T \) calculated each time, and record the value of \( T(s,\text{init}) \) in the position of \( dp[s][\text{init}] \).

The final solution result is as follows:

![Fig. 7 Dynamic programming method to solve a simple case result](image)

The total distance traveled by the traveling salesman is about 10127.55km, and the time required to solve the problem is about 0.02667s.

(4) The realization of genetic algorithm is similar to the biological evolution process in nature. First, it is necessary to set a coding scheme to "digitize" the potential solution to the problem, and then initialize the population. The digitized coding is the individual in the population. Then through decoding, the fitness of each individual is calculated, and individuals with high fitness are selected, and then the genes of these individuals are cross-mutated to produce offspring, and the above process is repeated through offspring.

The basic process of genetic algorithm is: Initialization; Individual evaluation; Selection; Crossover; Mutation; Termination condition.

Before running the program, set the parameters first. Set the cross-matching ratio to 0.8, the mutation rate to 0.02, the population size to 500 people, and the number of iterations to 500. A total of ten calculations are made, and the result of the shortest route is selected. The results of a simple case solved by genetic algorithm are shown in the figure below:

![Fig. 8 Genetic algorithm solving simple case results](image)

The total distance covered by the genetic algorithm is about 10127.55km, and the time required for the solution is about 0.0245s.

In order to further compare the solution performance in simple cases, 3 groups of 10 cities are added here. The characteristics of these 3 groups of cases are that the numerical calculation range involved is small and it is relatively easy to verify. Each case is solved by greedy method, branch and bound method, dynamic programming method and genetic algorithm respectively, and the total route length and solution time obtained by them are compared. The comparison results are shown in the following table. The order of the result graphs is from left to right, followed by greedy method, branch and bound method, dynamic programming method and genetic algorithm.
Table 2. Solving the results of the first group of 10 city cases (km, s)

| Method                | Length  | Time       |
|-----------------------|---------|------------|
| greedy method         | 22.37   | 9.19e-05   |
| branch and bound method| 21.54   | 2.77       |
| dynamic programming method| 21.54   | 0.027      |
| genetic algorithm     | 21.54   | 0.025      |

Table 3. Solving the results of the second group of 10 city cases (km, s)

| Method                | Length  | Time       |
|-----------------------|---------|------------|
| greedy method         | 21.97   | 0.00016    |
| branch and bound method| 21.19   | 13.16      |
| dynamic programming method| 21.19   | 0.025      |
| genetic algorithm     | 21.19   | 0.021      |

Table 4. Solving the third group of 10 city case results table (km, s)

| Method                | Length  | Time       |
|-----------------------|---------|------------|
| greedy method         | 23.05   | 0.00010    |
| branch and bound method| 21.63   | 1.52       |
| dynamic programming method| 21.63   | 0.027      |
| genetic algorithm     | 21.63   | 0.032      |

4. Conclusion
It can be seen from the above results that among the four different algorithms to solve the same TSP problem, although the greedy method requires the shortest time, most of the results obtained are inferior to other algorithms. The dynamic programming method, branch and bound method and genetic algorithm can all obtain the optimal solution. The difference is that the time required to solve the dynamic programming method and genetic algorithm is shorter. When calculating small-scale cases, as long as the genetic algorithm has reasonable parameters, sets a reasonable population size and enough iterations, the shortest path can also be found. As the number of iterations increases, the overall fitness obtained each time also increases. The route can gradually approach the optimal solution in the result.
References

[1] Li Qing, Wei Guangcun, Gao Lan, Qiu Guohua, Xiao Xinguang. Improved genetic algorithm for solving TSP problem [J]. Software Guide, 2020, 19(03): 116-119.

[2] Xu Jia, Han Fengqing, Liu Qixin, Xue Xiaoxia. A bioinformatics heuristic genetic algorithm for solving TSP [J/OL]. Journal of System Simulation: 1-10 [2021-08-27].

[3] Gao Shan. Research and Application of Solving TSP Based on Greedy Random Adaptive Grey Wolf Optimization Algorithm [D]. Taiyuan University of Technology, 2019.

[4] Xiao, Mingyu, Nagamochi, Hiroshi. An Exact Algorithm for TSP in Degree-3 Graphs Via Circuit Procedure and Amortization on Connectivity Structure [J]. Algorithmica, 2016, 74(2): 713-741.

[5] Jiangjiang Wang, Yanchao Lu, Ying Yang, Tianzhi Mao. Thermodynamic performance analysis and optimization of a solar-assisted combined cooling, heating and power system [J]. Energy, 2016, 115.