The Travelling Salesman Problem – The Testing of Various Methods

Seán Patterson (Nominee), Conor Moloney, Stephen Connolly

Abstract

The 'Travelling Salesman Problem' (TSP) has become one of the most well-known problems in operations research. The problem has become so well-known because in terms of optimally, it has proven deceptive difficult to solve. The problem states that given a list of cities and the distances between the cities, what is the shortest path that visits each city only once and returns to its starting point? There are two main considerations when attempting to solve the TSP; the effectiveness at finding the optimal solution and the time taken to compute that solution. This shows why the problem has become so famous. On the one hand we aim to find the optimal result, but we need to find it quickly. Finding a solution that is close to optimal with a low computational time is what we set out to test, using three heuristic methodologies; the nearest neighbour method, the penalty method and simulated annealing. It was found that simulated annealing was the best of the three tested methodologies at solving the TSP as the solutions generated were consistently lower that the other methods and thus can be considered closer to the optimum. Its computational time was slightly longer in comparison to the nearest neighbour method, which proved the best in times of computational time, however it was substantial more optimal in terms of solving the TSP .

1. Introduction

The 'Travelling Salesman Problem' (TSP) is an algorithmic problem in the field of operations research, primarily focused on optimisation. The problem affirms that given a finite number of cities (nodes), n, along with the distance between each city (node), we aim to find a solution that visits each of the cities (nodes) exactly once before returning to the starting point (city/node) (Carroll, 2019). Framed another way; the aim of solving the problem is to find a Hamiltonian cycle/ tour, whose total distance is minimal (Grotschel and Nemhauser. 2007).

The TSP has become one of the most widely known problems for those in operations research and computer science along with a host of other disciplines as finding an algorithm that definitively solves the problem has proven deceptively difficult. The reason for this is simply due to the sheer number of possible tours that can be attempted. For n number of cities, we have m number of edges or routes between each city, where m = n(n-1)/2. Given n connected by m edges, we can use probability to show that there are (n-1)!/2 unique tours that are possible (Carroll, 2019).

Given that as n increases, the number of unique tours grows exponentially, it becomes clear that we cannot simply test each tour to find the optimal outcome. As such, numerous methods have been put forward in an attempt to solve the problem using algorithms. These range from integer linear programming to variations on graph theory techniques to heuristics.

Heuristics is an approach to problem-solving that aims to use algorithms that search in an educated manner. Heuristic does this in a number of ways including using trial and error methods or using exploratory problem-solving techniques (Carroll, 2019). Specific to the TSP, heuristic techniques have been widely used as an approach to attempt to solve the problem. One such heuristic technique is called 'insertion heuristics' (Gendreau et al., 1998) which employs a method of searching for the 'nearest neighbour.' This method simply suggests that we first pick a start node/city, then visit its nearest neighbour and then repeat this process until all the nodes/cities are visited. This is very similar to Dijkstra's minimal spanning tree (MST) algorithm (Fredman et al., 1987) which states that we should first select any starting node, then select the node closest to it to join the spanning tree, then select the closest node not presently in the spanning tree ensuring no cycle is formed. This is repeated until all nodes join the spanning tree (Carroll, 2019). The only major difference between the 'nearest neighbour' heuristic insertion algorithm and Dijkstra's MST algorithm is that there is a final step required at the end to complete the tour.

Another heuristic approach that has been used to attempt to solve the TSP is called 'improvement heurisitcs' (Van Breedam, 1995) which eliminates two edges and reconnects the two resulting paths in a different way to obtain a new tour (Carroll, 2019).

Another heuristic technique is called 'simulated annealing.' This technique aims to use concepts employed in areas of physics such as thermodynamics (Cerny, 1985) to attempt to solve the TSP. In physics this technique is often used to find the lowest-temperature of a system or the formation of crystalline structures. It aims to use the metropolis algorithm and thus we can think of ‘simulated annealing’ as a sequence of metropolis algorithms (Carroll, 2019). It bears similarities to hill-climbing, but instead uses probabilistic methods to help avoid getting stuck in local optimum (a common issue with traditional hill-climbing techniques) and find global optimum instead.

A final heuristic technique to look at when solving the TSP is the penalty method as proposed by Amponsah et al. (Amponsah et al., 2016). This algorithm uses a node x node (n × n) cost matrix populated with the edge values (distance between two nodes). Using this matrix, the penalty algorithm aims to use subtraction so that a zero value can be found in every row or column. A penalty is then associated with each zero value and from this the matrix is gradually reduced by removing the zero values until no rows or columns remain. This method uses the Hungarian algorithm (Kuhn, 1955) as its basis but aims to improve upon it. The Hungarian algorithm adds or subtracts a constant to every element of a row or column of the cost matrix in a minimization model to create zeros in the given cost matrix. From this it then tries to complete the assignment using the zero values (Basirzadeh, 2014).

The aim of this paper is to test a number of the afore mentioned methods and techniques for solving the TSP to determine which provides the most optimised results based on a specific data set. Broadly speaking we will be testing three types of heuristic methodologies; the first will be insertion heuristics (nearest neighbour), the second will be the penalty method and the third will be simulated annealing.

2. Methodology

In the research for this paper we used several methods and/or techniques to attempt to solve the TSP to determine which provided the most optimised results based on the data set. For our data set we decided to use a 14-city problem for cities in Burma, as shown in appendix 1. The data set provided the x, y coordinates for each of the cities(nodes) and from this the distance between each of the cities(nodes) had to be calculated. To do this we calculated the Euclidean distance using the formula shown in figure 1. (The Euclidean distance is equivalent to the Pythagorean theorem). Once the distances(edges) had been calculated, a node x node (n x n) incidence matrix was formed with the distance/edge values (m) populating the body of the matrix, as shown in appendix 1.

$$d(a,b) = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)}$$

Figure 1: The equation for Euclidean distance between point a (x_1, y_1) and point b (x_2, y_2)

Before using the calculated n x n incidence matrix, we first had to set the parameters for the investigation. Given that the TSP aims to find the shortest Hamiltonian tour/cycle (Grotschel and Nemhauser, 2007), we can first determine that each node (vertex) must be visited exactly once. We can also clarify that the direction of travel is not restricted as shown by the equation $C_{ij} = C_{ji}$ and thus we have a symmetric TSP (Carroll, 2019). We can write this algebraically as shown below in figure 2; where C_{ij} represents the value of the distance between 2 nodes and x_{ij} represents the displacement vector (Amponsah et al., 2016).

Minimise:

$$W = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_{ij}$$

Subject to:

$$\sum_{i=1}^{n} x_{ij} = 1$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad x_{ij} \in [0,1]$$

$$x_{ij}\{1; if\ the\ i\ th\ node\ is\ assigned\ to\ the\ j\ th\ node\ or\ visa\ vera$$

$$0;\ Otherwise\}$$

$$C_{ij} = C_{ji}$$

Figure 2: Algebraic notation for the TSP

The first method we tested to solve the TSP was the insertion heuristic (Gendreau et al., 1998) technique of finding the ‘nearest neighbour’. This method can be broken down into 3 steps:

1. Pick a start node/city
2. Visit its nearest neighbour
3. Repeat step 2 until all the nodes/cities are visited (Carroll, 2019).

As previously mentioned, this method bares a number of similarities to Dijkstra’s MST algorithm (Fredman et al., 1987) which can be broken down into 4 steps:

1. Select any starting node
2. Select the node closest to it to join the spanning tree

3. Select the closest node not presently in the spanning tree ensuring no cycle is formed.
4. Step 3 repeated until all nodes have joined the spanning tree (Carroll, 2019).

Given that Dijkstra's MST algorithm specifies that we are to select the closest neighbour not previously selected, we decided to use this as the basis of forming a more complete definition of the insertion heuristic 'nearest neighbour' process. The steps we used are as follows:

1. Select any starting node/ city
2. Select the node closest to it to join the Hamiltonian tour
3. Select the closest node not presently in the Hamiltonian tour to ensure that no cycle is formed.
4. Repeat step 3 until all nodes have been selected.
5. Join the final selected node to the starting node to complete the tour.

The second method we tested to attempt to solve the TSP was the heuristic technique known as the penalty method as proposed by Amponsah et al. (Amponsah et al., 2016). This algorithm uses the n × n cost matrix populated with the edge values (distance between two nodes). The steps we employed were as follows:

1. Using the n x n matrix, the lowest value in each row was subtracted from the other values in the row.
2. Step 1 was repeated for the columns in the matrix until there was a zero value in every row and column.
3. A penalty was added to each zero value by adding the lowest value in its respective row to the lowest value in its respective column.
4. The highest resultant penalty was selected, and that zero value's column and row were removed and recorded as n city/node (row) going to n city/node (column) for the TSP, thus determining the vector.
5. This is repeated until no more row or columns remain.

As previously mentioned, the penalty method uses the Hungarian algorithm (Kuhn, 1955) as its basis but aims to improve upon it.

The third and final method we tested to attempt to solve the TSP was the heuristic technique known as the simulated annealing (Carroll, 2019). As previously stated simulated annealing is an optimisation technique that has similarities to hill-climbing, but which uses probabilistic methods to help avoid getting stuck in local optimum instead aims to find global optimum. The algorithm is similar to hill climbing but instead except of always picking the best move it picks a random move. If that move is better than the current solution it will always be accepted, if not there is still a chance that it will be accepted. This probability decreases exponentially with the "badness" of the proposed new solution i.e. how much worse (measured by delta E) it is than the previous solution. A parameter T (similar to temperature in an annealing system) will be used to help determine this probability. For higher values of T an inferior solution is more likely to be accepted. T starts high and is gradually decreased with each iteration of the algorithm. The number of iterations to take place is another important parameter which effects the usefulness of the algorithm.

3. Results

Our results for testing the different methodologies to solve the TSP indicated that the heuristic technique known as simulated annealing (Carroll, 2019) provided the most optimal results, particularly in comparison to the insertion heuristic technique of finding the nearest neighbour (Gendreau et al., 1998). Details of the comparison between these methods will be discussed later in the discussion section.

Using the insertion heuristic technique of finding the nearest neighbour (Gendreau et al., 1998) it was found that the optimal solution that we were able to obtain was 31.88, as shown in table 1. However, the average value is 36.38 and thus this shows a high level of discrepancy in results depending on which node/ city you choose as your starting point, suggesting that there are almost n number of different outcomes that can be obtained using this method. It should be noted that given the fact that we are moving from one city/ node to its nearest neighbour that this method follows a sequential pattern.

Start Node/ City	Nearest Neighbour - Total Distance
1	38.69
2	36.3
3	36.5
4	34.73
5	36.12
6	36.12

7	35.41
8	37.94
9	38.32
10	38.32
11	38.88
12	31.88
13	36.3
14	33.85

Table 1: The total value obtained for the TSP using the nearest neighbour method with each city/ node as our start point. The values were taken from the n x n matrix in appendix 1.

Our results when using the heuristic penalty method (Amponsah et al., 2016) showed that the optimal solution was 31.57. Using this method, it was suggested that we start at city/ node 13 and moved to city/ node 7. In terms of the n x n matrix; row 13 and column 7 were removed and the reduced matrix was formed. We continued in this manner until there were no rows and columns remaining (note that column 13 had to remain the final city/ node) and we had completed the Hamiltonian tour. The results are shown in table 3. An interesting observation that we noted when carrying out this method was that each reduction of the matrix did not follow a sequential pattern. For example, during the assignment of the first vector we moved from city/ node 13 to city/ node 7. However, during the assignment of the second vector we moved from city/ node 12 to city/ node 6 and not from city/ node 7 to another city/ node, as shown in table 3. This shows the effectiveness of the method/ algorithm as it is able to keep track of what has been previous used without breaking any of the constraints.

Based on Penalty 0s the selected edges are:	
n (row) going to n (column)	Distance
13 to 7	1.07
12 to 6	0.66
6 to 5	3.44
7 to 12	1.77
5 to 4	4.80
3 to 14	2.01
4 to 3	2.45
14 to 2	3.62
1 to 8	0.75
2 to 1	1.66
8 to 11	1.28
9 to 10	2.37
11 to 9	0.23
10 to 13	5.45
Total	**31.57**

Table 2: The total value obtained for the TSP using the penalty method.. The values were taken from the n x n matrix in appendix 1.

Our results when using simulated annealing showed that it was the most optimal in terms of minimizing the distance for the TSP. As mentioned previously T (parameter to determine the probability) starts high and is gradually decreased with each iteration of the algorithm. The optimal solution was found to be 30.88 using either node/ city 8, 10 or 14 as our starting point. It is worth noting that the average solution using this technique was 31.39 with the largest (least optimal) distance being 32.18. This seems to indicate that regardless of which note we start with it will find a more optimal solution that other methods.

Start Node/ City	Simulated Annealing – Total Distance
1	31.21
2	31.83
3	31.21
4	31.23
5	32.18
6	31.83
7	31.21
8	30.88
9	31.21
10	30.88
11	31.83
12	31.88
13	31.23
14	30.88

Table 3: The total value obtained for the TSP using simulated annealing with each city/ node as our start point. The values were taken from the n x n matrix in appendix 1.

4. Discussion

Comparing our results between the three afore mentioned methodologies to solve the TSP we can deduce that the heuristic technique known as the simulated annealing (Carroll, 2019) proved a more optimal algorithm/ method than both the insertion heuristic technique of finding the nearest neighbour (Gendreau et al., 1998) and the penalty method (Amponsah et al., 2016). All three methodologies worked by providing a valid solution to the TSP without violating any of the constraints. Thus we can state that each method found 1 of the 3,113,510,400 possible Hamiltonian tours ((n-1)!/2 = 13!/2). However, using simulated annealing proved the most optimal with a lowest value of 30.88.

The insertion heuristic technique of finding the nearest neighbour (Carroll, 2019) proved effective at solving the TSP without violating any of the constraints but was not optimal. However, we found that there was discrepancy between the results obtained using this method depending on which city/ node we chose to start with. These values ranged from 38.88 to 31.88. This shows that each city/ node when used as the starting city/ node will generate a different solution to the TSP, meaning that we may be able to generate n number of different solutions using this method and thus this method is not the most optimal to use in terms of minimizing distance as there is an element of trial and error as to twhich city is the best to start with. Despite this, there are benefits to using this method, primarily the computational time (Huang, W. and Yu, J. X., 2017). The reason that the time to compute the solution is relatively small in comparison to other methods to simply due to the fact that at each stage only one decision/ calculation is to be made, namely finding the shortest route/ path, provided it does not violate any constraint. For small values of n this method may be viewed as sub-optimal, however for very large values of n it may become useful due to the time it takes to obtain a valid solution.

The penalty method (Amponsah et al., 2016) proved both effective at solving the TSP without violating any of the constraints and at showing some level of optimality. The obtained value of 31.57 was smaller than the lowest obtained value using the nearest neighbour method of 31.88 showing that it came closer to minising the TSP and thus it came closer to the optimal solution. Amponsah et al. in their 2016 paper proposing the idea found this method to be more effective than the Hungarian algorithm (Kuhn, 1955) from which the method is derived. Their paper also seemed to suggest that as well as being more optimal that the Hungarian algorithm, it was also faster in relation to the computational time to solve the TSP. While we can consider this an effective approach to solve the TSP, we must mention that as n grows the time to compute the solution grows as well, not least due to the fact that the size of the matrix grows to n^2. There is more complexity to the algorithm itself compared to the nearest neighbour method. The nearest neighbour method simply looks for the smallest value, whereas the penalty method has to complete a number of steps that are similar in nature to a gaussian elimination before it can determine each vector. However, relative to other methods and algorithms the penalty method can still considered quick but could be improved.

Simulated annealing proved the most effective at solving the TSP with an optimal solution of 30.88. It also proved the most consistent with the solutions ranging from 32.88 to 30.88. A number of factors should be considered when implementing this algorithm, including number of iterations the algorithm should have, the starting value and alpha, the rate at which the value reduces. More iterations will generally improve the algorithms ability to identify the most optimum solution but at the cost of an increased run-time. The same is generally true of a higher starting value and lower alpha. This trade-off should be considered for any implementation of this algorithm, especially implementation on large data-sets where run-time may be an important factor. Comparing simulated annealing to the penalty method we can see that they both provide relatively optimised solutions, however, we can deduce that simulated annealing is more optimal as not only is its lowest value lower than the penalty method, but it provides us with more answers (one for each starting node/ city). When compared to the nearest neighbour method we can clearly see that simulated annealing is a more optimal approach to solving the TSP, as shown in table 4. On average simulated annealing improved the optimal solution found by 13.2% However, it should be noted that the computational time is longer for simulated annealing.

Start Node	Nearest Neighbour	Simulated Annealing	Improvement
1	38.69	31.21	0.193
2	36.3	31.83	0.088
3	36.5	31.21	0.145
4	34.73	31.23	0.101
5	36.12	32.18	0.109
6	36.12	31.83	0.119
7	35.41	31.21	0.114
8	37.94	30.88	0.186
9	38.32	31.21	0.186
10	38.32	30.88	0.194
11	38.88	31.83	0.182
12	31.88	31.88	0
13	36.3	31.23	0.139
14	33.85	30.88	0.088

Table 4: The total value obtained for the TSP using the nearest neighbour methd vs simulated annealing with each city/ node as our start point. The improvement for each node for simulated annealing is shown under improvement. The values were taken from the n x n matrix in appendix 1.

5. Conclusion

Based on the results of this investigation it can be deduced that simulated annealing has been proven to be the most effective of the three solutions to the TSP, particularly in terms of optimality. As an algorithm it is slightly more complex that other methodologies and potentially requires a larger number of iterations before the final solution is reached, which can slow its computational time slightly. However, this vastly improves its efficiency at finding the optimal solution. This algorithm could be further improved by tuning parameters such as number of iterations and starting value, although this tuning should also keep run-time in mind.

The nearest neighbour method provides us with a fast and valid solution to the TSP making it very valuable for large n values. We must take into consideration the fact that because of the speed of processing and the small number of steps to find a valid solution to the TSP that we give way to a degree of efficiency and thus the solution is can be considered sub-optimal. This can be attributed to the fact that at each stage the problem fails to consider the entire problem structure and instead on focuses on the decision at hand to allow for the faster computational time.

The penalty method also proved an effective method at solving the TSP, providing an optimal value close to that of simulated annealing. However, it must be noted that it did not solve for each starting node/city and thus only suggested one route. Another factor to consider is the complexity of the algorithm; a decent knowledge of matrices is needed to understand this method and it proved difficult to implement using python.

In conclusion it can be stated that in terms of finding the optimal solution to the TSP simulated annealing can considered as the more effective methodology/ technique. It allows us to find a shorter Hamiltonian tour in relatively small amount of time. In terms of computational time only we can state that the nearest neighbour method is the more efficient though it does give way to optimally of the solution itself.

References

Amponsah, S., Otoo, D., Salhi, S. and Quayson, E. (2016). *Proposed Heuristic Method for Solving Assignment Problems*, American Journal of Operations Research, 6, 436-441

Basirzadeh, H. (2014). *Ones Assignment Method for Solving Traveling Salesman Problem*, Journal of mathematics and computer science 10 (2014), 258-265

Carroll, P. (2019). *Graph Theory and Networks*, lecture notes, Optimisation in Business MIS41160, University College Dublin, delivered October to November 2019

Carroll, P. (2019). *Introduction to Heuristics*, lecture notes, Optimisation in Business MIS41160, University College Dublin, delivered November 2019

Carroll, P. (2019). *Numerical Algorithms*, lecture notes, Optimisation in Business MIS41160, University College Dublin, delivered October to November 2019

Cerny, V. (1985). *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, Journal of Optimisation Theory and Applications, Vol. 45, No. l

Cormen, Thomas, H., Leiserson, Charles, E., Rivest, Ronald, L., Stein, Clifford (2001). *Section 24.3: Dijkstra's algorithm, Introduction to Algorithms*, MIT Press and McGraw–Hill, pp. 595–601.

Fredman, Lawrence, M., Tarjan, Robert, E. (1987). *Fibonacci heaps and their uses in improved network optimization algorithms*, Journal of the Association for Computing Machinery, 34 (3): 596–615.

Gendreau, M., Hertz, A., Laporte, G., Stan, M. (1998). *A Generalized Insertion Heuristic for the Traveling Salesman Problem with Time Windows*, Operations Research, Vol 46, No. 3, pp 293-432

Grotschel, M., Nemhauser, G. L. (2007). *George Dantzig's contributions to integer programming*, Discrete Optimization 5 (2008) 168-173, 169

Huang, W. and Yu, J. X. (2017). *Investigating TSP Heuristics for Location-Based Services*, Data Science and Engineering March 2017, Volume 2, Issue 1, pp 71–93

Kuhn, H. W. (1955). *The Hungarian Method for the assignment problem*, Naval Research Logistics Quarterly, 2: 83–97

Van Breedam, A. (1995). *Improvement heuristics for the Vehicle Routing Problem based on simulated annealing*, European Journal of Operational Research, Vol 86, Issue 3, pp 480-49

Appendix 1

Node	Coordinates	
	x	y
1	16.47	96.10
2	16.47	94.44
3	20.09	92.54
4	22.39	93.37
5	25.23	97.24
6	22.00	96.05
7	20.47	97.02
8	17.20	96.29
9	16.30	97.38
10	14.05	98.12
11	16.53	97.38
12	21.52	95.59
13	19.41	97.13
14	20.09	94.55

Appendix 1, table 1: x, y coordinates for each of the 14 cities

Incident Matrix - n x n (node x node) - Note: Body of the Matrix contains the values of distances between nodes (m)														
Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0.00	1.66	5.08	6.52	8.83	5.53	4.10	0.75	1.29	3.15	1.28	5.08	3.12	3.94
2	1.66	0.00	4.09	6.02	9.20	5.76	4.76	1.99	2.94	4.40	2.94	5.18	3.98	3.62
3	5.08	4.09	0.00	2.45	6.96	4.00	4.50	4.73	6.15	8.22	6.01	3.37	4.64	2.01
4	6.52	6.02	2.45	0.00	4.80	2.71	4.12	5.96	7.29	9.60	7.10	2.38	4.80	2.59
5	8.83	9.20	6.96	4.80	0.00	3.44	4.77	8.09	8.93	11.21	8.70	4.06	5.82	5.80
6	5.53	5.76	4.00	2.71	3.44	0.00	1.81	4.81	5.85	8.22	5.63	0.66	2.81	2.43
7	4.10	4.76	4.50	4.12	4.77	1.81	0.00	3.35	4.19	6.51	3.96	1.77	1.07	2.50
8	0.75	1.99	4.73	5.96	8.09	4.81	3.35	0.00	1.41	3.64	1.28	4.38	2.36	3.37
9	1.29	2.94	6.15	7.29	8.93	5.85	4.19	1.41	0.00	2.37	0.23	5.52	3.12	4.73
10	3.15	4.40	8.22	9.60	11.21	8.22	6.51	3.64	2.37	0.00	2.59	7.89	5.45	7.02
11	1.28	2.94	6.01	7.10	8.70	5.63	3.96	1.28	0.23	2.59	0.00	5.30	2.89	4.55
12	5.08	5.18	3.37	2.38	4.06	0.66	1.77	4.38	5.52	7.89	5.30	0.00	2.61	1.77
13	3.12	3.98	4.64	4.80	5.82	2.81	1.07	2.36	3.12	5.45	2.89	2.61	0.00	2.67
14	3.94	3.62	2.01	2.59	5.80	2.43	2.50	3.37	4.73	7.02	4.55	1.77	2.67	0.00

Appendix 1, matrix 1: Matrix containing each of the distances between the respective nodes. These values were used to as the data for the TSP and the respective methods to solve.