Travelling salesman problem

Abstract

With a rise of on – demand deliveries, modern businesses are obsessed with the increasing cost. Cost savings are essential; however, it is not the only thing that sets us apart from our competition. Speed, agility and transparency in the delivery process are the most important elements in logistics and supply chain. These factors enhance the customer experience. Finding the shortest route between multiple destinations is vital to develop this customer requirement. The “Traveling Salesman Problem” is coming up with this issue.

Introduction

The travelling salesman problem was mathematically formulated in the 1800s by the mathematician William Rowan Hamilton and by the British mathematician Thomas Kirkman. It was first considered mathematically in the 1930s by Merrill Meeks Flood who was looking to solve a school bus routing problem. Hassler Whitney at Princeton University generated interest in the problem, which he called the “48 states problem”. The earliest publication using the phrase “Travelling salesman problem” was the 1949 RAND Corporation reported by Julia Robinson,” On the Hamiltonian game ( a travelling salesman problem).”

Statement of Travelling Salesman Problem

The travelling salesman problem (also called the travelling salesperson problem or TSP) requires answers for the following question: “Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?”

As a graph problem

TSP can be modeled as an [undirected weighted graph](https://en.wikipedia.org/wiki/Graph_(discrete_mathematics)), such that cities are the graph's [vertices](https://en.wikipedia.org/wiki/Vertex_(graph_theory)), paths are the graph's [edges](https://en.wikipedia.org/wiki/Glossary_of_graph_theory_terms), and a path's distance is the edge's weight. It is a minimization problem starting and finishing at a specified [vertex](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) after having visited each other [vertex](https://en.wikipedia.org/wiki/Vertex_(graph_theory)) exactly once. Often, the model is a [complete graph](https://en.wikipedia.org/wiki/Complete_graph) (i.e., each pair of vertices is connected by an edge). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour.

Asymmetric and symmetric

In the symmetric TSP, the distance between two cities is the same in each opposite direction, forming an [undirected graph](https://en.wikipedia.org/wiki/Undirected_graph). This symmetry halves the number of possible solutions. In the asymmetric TSP, paths may not exist in both directions or the distances might be different, forming a [directed graph](https://en.wikipedia.org/wiki/Directed_graph). [Traffic collisions](https://en.wikipedia.org/wiki/Traffic_collision), [one-way streets](https://en.wikipedia.org/wiki/One-way_traffic), and airfares for cities with different departure and arrival fees are examples of how this symmetry could break down.

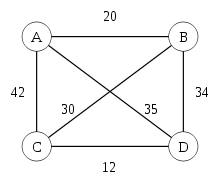


Figure 1: Symmetric TSP with four cities.

Travelling salesman problem example

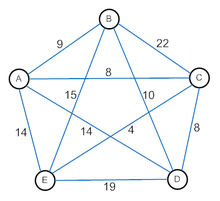


Figure 2: TSP problem

There are four steps to solve the problem

1. Start with an arbitrary node (first node).
2. From the current node, find a smallest collecting edge to a new node. Mark the new node as selected node.
3. Keep doing step 2 until there is no selected node.
4. Back to first node.

The problem starts with 5 cities and the distance between them. We now apply the “nearest neighbor algorithm”, start from each node and then find out which way should be the shortest way for the salesman.

Let’s assume A is the first city and the salesman has to arrive every remaining city.

From A, nearest city is C, length AC=8;

C now is the new selected node.

From C, nearest city which have been not arrived is E, length CE=4;

E now is the new selected node.

From E, nearest city which have not been arrived is B, length EB=15;

B now is the new selected node.

From B, nearest city which have not been arrived is D, length BD=10;

D now is the new selected node.

Every city now has been arrived, so from D, the salesman will come back to A: length DA=14;

The total length ACEBDA is

8+4+15+10+14=51

Redo four steps with other cities (not A):

|  |  |  |
| --- | --- | --- |
| Starting city | Route | Length |
| B | BACEDB | 50 |
| C | CEABDC | 45 |
| D | DCEABD | 45 |
| E | ECABDE | 50 |
| E | ECDBAE | 45 |

Table 1: Router calculation.

Conclusion: There are three shortest way for the salesman. If the starting city is A, the best way is ABDCEA=45.

Harvesine formula

The haversine formula determines the [great-circle distance](https://en.wikipedia.org/wiki/Great-circle_distance) between two points on a [sphere](https://en.wikipedia.org/wiki/Sphere) given their [longitudes](https://en.wikipedia.org/wiki/Longitude) and [latitudes](https://en.wikipedia.org/wiki/Latitude). Important in [navigation](https://en.wikipedia.org/wiki/Navigation), it is a special case of a more general formula in [spherical trigonometry](https://en.wikipedia.org/wiki/Spherical_trigonometry), the law of haversines, which relates the sides and angles of spherical triangles.

Formulation

Let the [central angle](https://en.wikipedia.org/wiki/Central_angle)  between any two points on a sphere is:

Where:

is the distance between the two points along a great circle of the sphere.

is the radius of the sphere.

The Haversine formula allows the [Haversine](https://en.wikipedia.org/wiki/Haversine_function" \o "Haversine function) of  (that is, hav()) to be computed directly from the latitude and longitude of the two points:

.

Where:

are the latitude of point 1 and latitude of point 2( in radians),

are the longitude of point 1 and longitude point 2( in radians).

Problem

Let’s assume that a salesman has to travel 10 biggest cities of Germany (or USA). Our mission is trying to help him to find out which is the shortest way when he is in a particular city.

|  |  |  |
| --- | --- | --- |
| City | Latitude | Longitude |
| Berlin Capital | 52.520008 | 13.404954 |
| Cologne | 50.935173 | 6.953101 |
| Dortmund | 51.514244 | 7.468429 |
| Dusseldorf | 51.233334 | 6.783333 |
| Essen | 51.450832 | 7.013056 |
| Frankfurt | 50.110924 | 8.682127 |
| Hamburg | 53.551086 | 9.993682 |
| Munich | 48.137154 | 11.576124 |
| Kamp-Lintfort | 51.502530 | 6.516200 |
| Stuttgart | 48.783333 | 9.183333 |

Table 2: 10 biggest cities in Germany.

We now apply Haversine formula to Python to resolve this problem.

Result and discussion

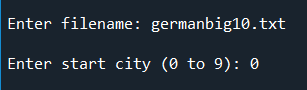


Figure 3: Needed input.

Discussion: In order to run the code, we have to execute a file that includes information about all 10 cities. For example, the file is (germanbig10.txt). And if the salesman starts at Berlin Capital, we set the starting city equals 0.

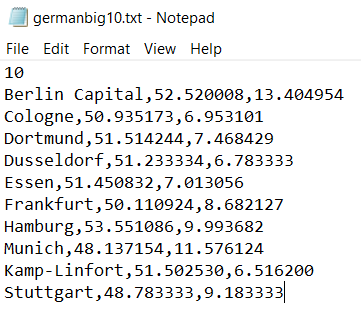


Figure 4: File input.

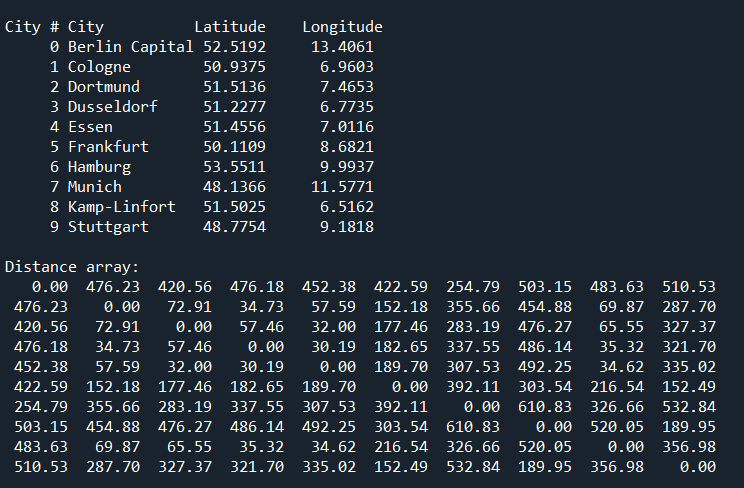


Figure 5: List of cities and 2D distance array.

Discussion: After running the file, list of cities and 2D distance array will be represented. By applying Harversine formula, the distance from Berlin Capital to other cities is calculated and show in the first row. Noted that if the salesman starts from Cologne, the second row will show distance between Cologne and others, etc.

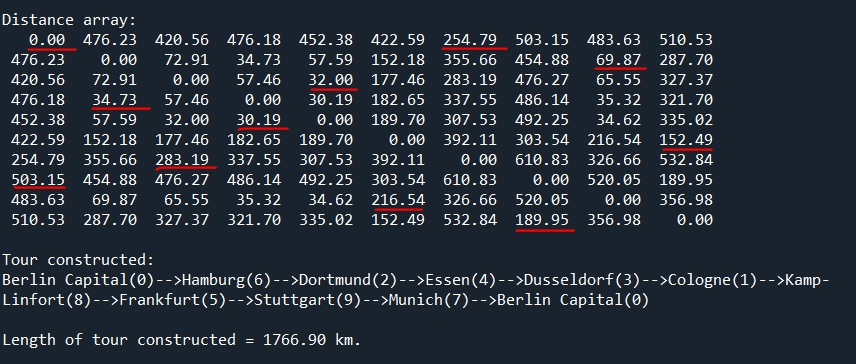


Figure 6: Tour constructor and total distance.

Discussion: Apply “nearest neighbor algorithm”, we will have a “Tour constructed” that is exactly is the shortest way for the salesman (figure 6- Distance array – highlighted numbers):

|  |  |  |
| --- | --- | --- |
| Line | Route | Distance (km) |
| 0 | Berlin Capital → Hamburg | 254.79 |
| 6 | Hamburg→ Dortmund | 283.19 |
| 2 | Dortmund→ Essen | 32.00 |
| 4 | Essen→ Dusseldorf | 30.19 |
| 3 | Dusseldorf → Cologne | 34.73 |
| 1 | Cologne → Kamp-Lintfort | 69.87 |
| 8 | Kamp-Linfotrt→ Frankfurt | 216.54 |
| 5 | Frankfurt→ Stuttgart | 152.49 |
| 9 | Stuttgart→ Munich | 189.95 |
| 7 | Munich→ Berlin Capital | 503.15 |
|  | Total | 1766.9 |

Table 3: The shortest route from Berlin.

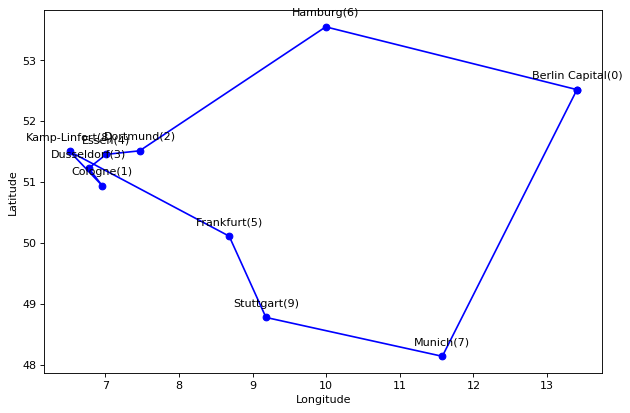


Figure 7: Route Graph.

Conclusion and application.

Conclusion

As we can see, the “Travelling salesman problem” can be solved by using Harvensine formula. It is cannot be denied that Harvensine does not work exactly in real life [because Harvensine formula is calculated by optimization, that’s mean there are no obstacles between two points (skyway)]. However, it provides a complete aspect of the solution for the problem and assist airlines to find out the shortest route for each flight.

Application

TSP has a number of applications even in its purest formulation, such as planning, logistics and chip manufacturing. Slightly modified, it appeared as a side problem in many areas, such as DNA sequencing. In these applications, the conceptual city represents, for example, the customer. The weld point or the DNA fragment, and the conceptual distance represent travel time or cost, or a similar measure between segments TSP also appears in astronomy, because astronomers observing multiple sources will want to minimize the telescope travel time between sources; In such problems, the TSP can be embedded with an optimal control problem. In many applications, additional constraints such as resource constraints or a time window be imposed.

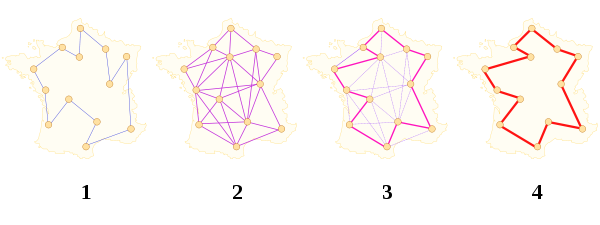


Figure 8: Ant colony optimization (example).

Reference

Dorigo, Marco; Gambardella, Luca Maria (1997). "Ant Colonies for the Traveling Salesman Problem". Biosystems. 43 (2): 73–81. [CiteSeerX](https://en.wikipedia.org/wiki/CiteSeerX_(identifier)) [10.1.1.54.7734](https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.54.7734). [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1016/S0303-2647(97)01708-5](https://doi.org/10.1016%2FS0303-2647%2897%2901708-5). [PMID](https://en.wikipedia.org/wiki/PMID_(identifier)) [9231906](https://pubmed.ncbi.nlm.nih.gov/9231906). [S2CID](https://en.wikipedia.org/wiki/S2CID_(identifier)) [8243011](https://api.semanticscholar.org/CorpusID:8243011)

["Search for "Traveling Salesperson Problem""](https://scholar.google.com/scholar?q=%22traveling+salesperson+problem%22). Google Scholar. Retrieved 23 November 2019.

Ross, I. M.; Proulx, R. J.; Karpenko, M. (6 May 2020). "An Optimal Control Theory for the Traveling Salesman Problem and Its Variants". [arXiv](https://en.wikipedia.org/wiki/ArXiv_(identifier)):[2005.03186](https://arxiv.org/abs/2005.03186)

Ross, I. M.; Proulx, R. J.; Karpenko, M. (July 2019). ["Autonomous UAV Sensor Planning, Scheduling and Maneuvering: An Obstacle Engagement Technique"](https://dx.doi.org/10.23919/acc.2019.8814474). 2019 American Control Conference (ACC). IEEE: 65–70. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.23919/acc.2019.8814474](https://doi.org/10.23919%2Facc.2019.8814474). [ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)) [978-1-5386-7926-5](https://en.wikipedia.org/wiki/Special:BookSources/978-1-5386-7926-5). [S2CID](https://en.wikipedia.org/wiki/S2CID_(identifier)) [201811463](https://api.semanticscholar.org/CorpusID:201811463)

[U. S. Census Bureau](https://en.wikipedia.org/wiki/U._S._Census_Bureau) Geographic Information Systems FAQ, (content has been moved to [What is the best way to calculate the distance between 2 points?](http://www.movable-type.co.uk/scripts/GIS-FAQ-5.1.html))

[The Traveling Salesman Problem Home](http://www.math.uwaterloo.ca/tsp/) – ausführliche Informationen zum Traveling Salesman Problem (englisch)

https://thegpscoordinates.net/ list of countries, distance calculator and converter.