# 1 The Traveling Salesperson Problem

The TRAVELING SALESPERSON PROBLEMS asks, given a complete, directed graph with real-valued edge lengths, what is the minimum length cycle that visits every node exactly once? This classic computer science problem is NP-hard. It is the problem most computer scientists reach to when explaining hardness to non-computer scientists since it is easy to motivate with stories of UPS drivers.

In its most general version, TSP is very hard to approximate. In fact, one can show it is hard to approximate up to any arbitrary factor. This is because TSP is a generalization of HAMILTONIAN CYCLE and since one can view the lack of edge in the HAMILTONIAN CYCLE graph as an edge of length $\infty$ in the corresponding TSP graph, there is an arbitrarily large gap between instances where there is a Hamiltonian cycle (optimal is length $n$ in the TSP) and there is no Hamiltonian cycle (optimal has length $\infty$ in the TSP). Notice here that *any* polynomial-time approximation algorithm would be able to detect this difference!

## 1.1 Metric TSP

Because generalized TSP is very hard to approximate, and because most real-world instances of TSP have more structure, we often restrict our attention to subclasses of the problem. One popular subclass is METRIC TSP. The metric version assumes the graph can be embedded in some metric space. This just means that

- the distances between nodes are symmetric (so the graph is essentially undirected),
- the edge-lengths are positive, although the distance between any node and itself is always 0,
- and the triangle inequality holds.

The metric version of TSP has an easy 2-approximation:

1. Fnd a minimum spanning tree for the graph. The cost of the minimum spanning tree is a natural lower bound on the minimum length tour.
2. Trace the edges of the tree. This yields a sequence of nodes $v = v_1, \ldots, v_{2n-2}$ where $v_{2n-2} = v_1$. Notice this path has length at most twice the length of the minimum spanning tree. Hence it has length at most twice the cost of the minimum length tour.
3. Remove any node from $v$ if it appears earlier in the sequence. This yields a well-formed tour. Furthermore, the cost could not have increased since the triangle-inequality holds.

Christofides developed a 3/2-approximation that essentially throws the kitchen sink at the problem:

1. Find a minimum spanning tree $T$ for the graph $G$. Note that every tree has an even number of odd-degree nodes.
2. Consider the complete subgraph of $G$ over all the odd-degree nodes from $T$. Find a minimum weight perfect matching $M$ on this graph.
3. Combine the matching and the spanning tree to form a multigraph[1] $H$.
4. Every node in $H$ has even-degree, so, by definition, it has an Euler tour. Call the tour $v = v_1, \ldots, v_t$.
5. The cost of $v$ is at most the cost of $T$ and the cost of $M$. We know the cost of $T$ is a lower bound on the cost of the optimal tour. What about $M$?
6. Since one can think of the optimal tour on $G$ as two *interleaved* perfect matchings, a minimum cost perfect matching is at most half the cost of the optimal tour.

[1] Edges that appear in both the matching and the MST appear twice now.

7. Thus, $v$ is a factor of 3/2 larger than the optimal TSP tour. Removing the redundant vertices from $v$ (as we did with the trace of the MST before) leaves us with a valid TSP tour.

Christofides' algorithm is the best known for metric TSP[2]. There is no PTAS for metric TSP (unless P=NP) but there is still a gap between the upper and lower bounds of the approximation ratio.

### 1.2 Euclidean TSP

Euclidean TSP is the TSP problem where the nodes are points in the $d$-dimensional Euclidean plane. The distance between any pair of points is their Euclidean distance. Since Euclidean space is a metric space, we know that Euclidean TSP has a 3/2 approximation. However, the lower bounds do not necessarily hold. Sanjeev Arora thought that they might hold, so he set about proving that Euclidean TSP does not have a PTAS. Surprisingly, the pursuit of this negative result, led him to a positive result: a polynomial-time approximation scheme for Euclidean TSP. The algorithm involves recursively decomposing the Euclidean space into quadrants until each quadrant contains only one point. Then a combination of dynamic programming and rounding yields the solution. Arora's original algorithm ran in time $n^{O(1/\epsilon)}$ in two-dimensions. One year later, he developed an algorithm running in time $n(\log n)^{O(1/\epsilon)}$. We will read the journal version of (essentially) the second paper.

## 2 Reading

Begin by reading

> *Polynomial Time Approximation Schemes for Euclidean Traveling Salesman and other Geometric Problems. Sanjeev Arora. Journal of the ACM. 45:5. 1998. 753–782*

You should feel free to skip section 3. Don't worry if you don't follow everything right away—there are many great lecture notes available on this topic. They will help fill in the gaps. I have notes from both Shuchi Chawla's course at Wisconsin and Michel Goemans' course at MIT linked on the website. Shuchi's notes follow the journal version a little more, but Goemans' notes are better written. In addition, there is a detailed set of slides on the website.

## 3 Questions

Several of this week's questions require a little research. Use whatever you can get your hands on, but don't regurgitate what you read directly — explain everything in your own words. Also, make sure to properly cite your sources.

**Question 1.** *Where do we use the metric space assumptions in the 2- and 3/2-approximations given above?*

**Question 2.** *Can you think of a metric space that isn't a Euclidean space? Feel free to use the web as a resource. Make sure to explain your answer.*

**Question 3.** *How practical is Arora's algorithm? You may find it useful to check out the following websites which are devoted to solving* TSP*:*
`http://www.research.att.com/~dsj/ctsp/`
`http://www.tsp.gatech.edu/`
*As well as the book:*
The Traveling Salesman Problem: A Computational Study by Applegate, Bixby, Chvatal, and Cook
*which is on reserve in Schow.*

**Question 4.** *Provide a short review of Arora's TSP article. What parts of the article did you enjoy the most? What parts did you least enjoy?*

[2] And is beautiful!

**Problem 1.** *Give a formal proof of my claim in Section 1 that general TSP has no good approximation algorithm. In other words, show that, for any positive function $f(n)$, there is no $f(n)$-approximation for TSP where $n$ is the number of nodes.*

**Problem 2.** *Show that the analysis of Christofides' algorithm is tight. In other words, describe a class of instances where Christofides' algorithm returns a solution that is exactly a factor of 3/2 larger than optimal.*

# 4 Theory versus Practice

**Problem 3.** *This problem asks you to empirically evaluate the 2-approximation for the TSP. You will also compare the 2-approximation against an algorithm that you develop.*

- *Begin by implementing the 2-approximation described in Section 1.1 using a language of your choice. If you choose Java, then you might find the* `JGraphT` *library useful. If you use C++, then check out the Boost graph library. Both libraries provide flexible graph representations along with algorithms for finding minimum spanning trees.*
- *Test your 2-approximation on at least 5 different instances of the traveling salesperson problem taken from the* symmetric traveling salesman problem *section of TSPLIB. The TSPLIB is available online at:* `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`*. All of the symmetric TSP instances have been solved, so you can truly compare the solution returned by the 2-approximation with the optimal solution.*
- *With your partner, develop your own polynomial-time heuristic algorithm for TSP. Borrow elements from Arora, Christofides, or any other algorithm that you read about. Can you beat the 2-approximation on the instances you tested against above? Make sure to describe your algorithm in prose and be prepared to give a high-level description of it during our tutorial session. Make sure to have fun.*