Euclidean Traveling Salesman Problem

Prakhar Mittal (UFID- 3909-9969)

Abstract:

This paper describes a survey on the Euclidean version of the very popular Traveling Salesperson Problem and includes a review and comparison of various approaches used to tackle such a problem. Additionally, in this paper, we will explore the significance of this simple, yet complex problem.

Introduction:

The Traveling Salesman Problem (TSP) is the best known fundamental node covering problem. Let us first try to understand what exactly TSP is, so then we can try to analyze its significance and why it is NP-hard. Consider a graph denoting each city as a vertex, where we try to visit every vertex at least once. To accomplish this, we start at a vertex (i.e. city) and visit every single one until we arrive at the starting vertex again. Our primary goal is to minimize the distance or time (whichever metric we opt for) by the salesman over the course of the journey. In other words, A weighted graph G with n vertices and we must find a path with a minimum cost that visits each vertex G exactly once (Hamiltonian Tour).[1]

As stated earlier, the TSP is NP-hard as are approximation algorithms for any constant approximation is also known to be NP-hard. Since it is known that there exists a polynomial-time reduction from the Hamiltonian Cycle to the Traveling Salesman Problem, it proves that TSP is NP-hard as well. For the Traveling Salesman Problem, we impose a metric (hence categorized as Metric TSP) which includes conditions a) that the distance between vertices is always greater than 0, b) one can travel unidirectional in the graph, c) finally, the most significant is having the triangle inequality, which, in simple terms, means that the sum of two distances between vertices in a triangle will always be more than or equal to the distance of the third side. However, the Metric TSP is NP-hard as well. Some examples of metric TSP's include Euclidean TSP, Rectilinear TSP, and the maximum metric. A practical example of the latter two is visible in a routing machine that drills holes in a circuit board[7]. In this paper, as stated, we look into a more particular case of Metric TSP in the form of Euclidean TSP.

Euclidean TSP:
Given: n points in $\mathbb{R}^2$ with euclidean distances, *i.e.*, $d(x,y) = \|x - y\|_2$.
Find: shortest tour that visits all points.

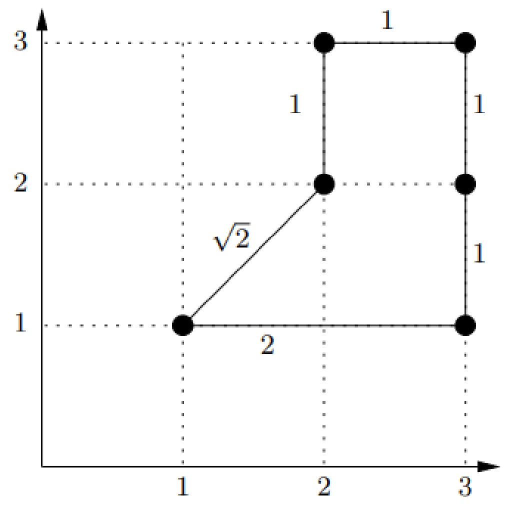

Fig.1[8] Example of an Euclidean TSP with tour of cost 6 + √ 2.

Let us try to study Euclidean (or “Geometrical”) TSP in detail and try to understand the importance in practical aspects even though it is known to be NP-Hard. Euclidean TSP follows the rule that the input graph G lies in a Euclidean plane and the costs between any two vertices are their Euclidean distances. We now proceed to examine and compare different popular approaches to this problem and try to study them based on the approximation schemes they propose.

Algorithm/Scheme 1:

In the chronology of studying polynomial-time approximation scheme for Euclidean TSP, we take a look at Christofides’ algorithm which guarantees a 3/2-approximation algorithm. The algorithm primarily involves two steps: 1) define the MST of graph T= (V, E), 2) evaluate M which is the cheapest cost matching the odd degree vertices in the MST hence making every other vertex to have an even degree (otherwise find M for the odd-degree vertices of MST).

Then, add the two and correspondingly find the Euler tour. Let OPT be the upper bound of the optimal cost. We obtain the optimal tour for the MST by removing an edge from it that costs less than OPT to get $\sum_{e \in M} c_e \leq$ OPT. Hence, we conclude the analysis after proving $\sum_{e \in M} c_e \leq$ OPT/2[5]. In the figure below, we can see the optimal tour of two perfect matchings of the odd degree vertices.

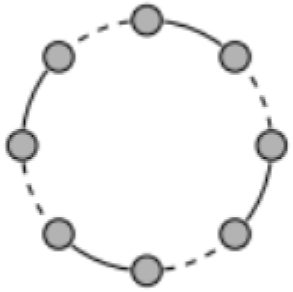

Fig. 2 [5]

Since M was the minimum matching, we can now say that one of the matching has a cost of at most OPT/2 and we finally get $\sum_{e \in M} c_e \leq$ OPT/2[5].
The following is the corresponding algorithm according to Wikipedia[7]:

1. Create a minimum spanning tree *T* of *G*.
2. Let *O* be the set of vertices with odd degree in *T*. By the handshaking lemma, *O* has an even number of vertices.
3. Find a minimum-weight perfect matching *M* in the induced subgraph given by the vertices from *O*.
4. Combine the edges of *M* and *T* to form a connected multigraph *H* in which each vertex has even degree.
5. Form an Eulerian circuit in *H*.
6. Make the circuit found in previous step into a Hamiltonian circuit by skipping repeated vertices (*shortcutting*).

Algorithm/Scheme 2:

In the general case, TSP was proved to be non-approximable until the 3/2 approximation scheme using the triangle inequality discussed in the Christofides algorithm discussed above. Now, we discuss the approach given by Grigni Koutsoupias, and Papadimitriou in “An Approximation Scheme for Planar Graph

TSP” showing that a Polynomial Time Approximation Scheme exits if the metric is the shortest-path metric of an unweighted planar graph (that is, all edge costs are one). The approximation ratio in $nO^{(1/\varepsilon)}$ runtime is $(1+\varepsilon)$ having $\varepsilon > 0$ is any constant[4]. In other words, consider a planer graph G with n number of points and any constant $\varepsilon > 0$. The algorithm uses a combination of divide & conquer and dynamic programming.

The algorithm is roughly divided into two stages/steps namely decomposition and an approximation stage. In the decomposition step, a decomposition tree with subgraphs having $O((\log n)/\varepsilon)$ constraint points of the given graph G. This is done by first parameterizing leaves of G as T. As roots of T are G itself, we declare G1 and G2 as children of G in T. After recursively constructing this, we define a graph H in the tree. Hence after that, the leaves in T are limited by the number of children in T’ which is more than or equal to the D times the children of T’[3]. Therefore, this stage is independent of ε and almost in linear time and hence certainly in polynomial time.

In the second step, which is the approximation stage, evaluate the values of $n^{O(1/\varepsilon)}$ approximated solutions in the bottom-up order. All the values are calculated using the values of the two children's subgraphs. Each of the table values is computed from the values of the tables of both their child's subgraphs.[3]
Let us now look at the analysis of this. As the running time has already been established to be $n^{O(1/\varepsilon)}$, we investigate the three sources of error to bound namely: Leaf error, Patching error and Contraction error. After carefully considering them all, we find that the total error after the addition is maximum of εn^3.

Algorithm/Scheme 3:

This algorithm provides us PTA scheme for Euclidean TSP in certain limited dimensions. Formally according to the paper: For every fixed $c > 1$ and given any n nodes in R^2, a randomized version of the scheme finds a $(1+1/c)$-approximation to the optimum traveling salesman tour in $O(n(\log n)^{O(c)})$ time[1]. Hence, theorem stated here paper provides a technique to break the Traveling Salesman Problem into small versions which are independent and that can make parallelizing the existing program.

The primary idea of this scheme is to do partitioning of the instance recursively. The algorithm works in the following way: At the starting, we perform ‘perturbation’ to make it well rounded. Briefly speaking, this means to make the coordinates integral As well as make the internode distance to be at least 8[1]. This can be achieved by keeping all points closest to their closest grid point after rescaling them. The paper defines the smallest square axis-aligned perturbed instance as the bounding box and length of the square to be L. hence whenever we say L, we talk about the size of the box. In the next step, we perform a dissection by partitioning into smaller squares with every square having 4 children squares. Hence, it can be viewed as a 4-ary tree. The root of this being the bounding box[1]. Thus, we now arrive at the state we apply Dynamic programming to evaluate the optimal tour, but before that, we specify some more terms. First, we allow solutions having edges to be ‘bent’ which help in easing out a Dynamic programming approach. Portals are those places where the edge of the square enters and leaves, which are m and equally spaced along the square's boundaries. Consider r to be the frequency of a tour crossing each square boundary. Hence, the optimal (m, r)-light portal tours are evaluated in a bottom-up approach of the 4-ary tree using Dynamic Programming.

Therefore, the cumulative result of the tour with the cheapest tours in the 4 squares to finally find the cheapest tour of the parent square. In the final step, with the original instance the optimal (m, r)-light portal-respecting tour is converted w.r.t the original instance.

Proof:

Let the state A[s,(s1, t1), . . . ,(s`, t`)] belong to square. Let path’s entry and exit locations be (s1, t1) to (s`, t’)[8].

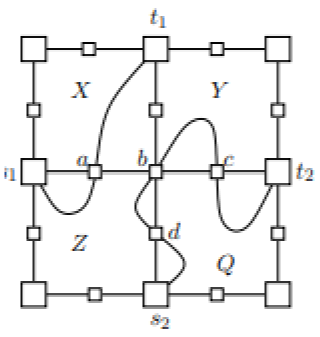

Fig 3[8]

For the optimal tour, we follow the example where we use values of A in the subsquare: A [s,(s1, t1),(s2, t2)] = A [X,(a, t1)] + A [Y,(b, c)] + A [Z,(s1, a),(d, b)] + A[Q,(s2, d),(c, t2)][8]. The computation time for all the entry into A will take polynomial time as there is a polynomial amount of nodes present in the quadtree. Therefore the dynamic program will run in polynomial time.

Conclusion:

In this short survey, we learned about the very popular traveling salesman problem, and learned about multiple techniques adopted in the quest to obtain a polynomial-time approximation of the problem.

First, we discussed what TSP is and about the special cases of Metric and Euclidean TSP. In Algorithm 1, we discussed the approximation algorithm by Christofides that runs in polynomial-time of Metric TSP also computes the tour having 3/2 times of OPT.

In Algorithm 2, we discussed the algorithm given by Grigni, Koutsoupias, and Papadimitriou[3] who had questioned the existence of PTA for Euclidean TSPs[4]. However, they prescribed the idea to design a PTA for the case having a shortest-path metric of a weighted planar graph.

Finally, in Algorithm 3, we discuss a very popular algorithm in the shape of Arora’s PTAS which comprised of two primary steps, which were finding the (m,r)-light portal respecting tour followed by shifted dissection. Interestingly, around the same time, Mitchell showed PTAS also recursively partitions the planar Euclidean TSP into subpartitions that can be solved using dynamic programming.[2]

References:

1. Arora, S. (1998). Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. Journal of the ACM (JACM), 45(5), 753–782. doi: 10.1145/290179.290180

2. Mitchell, J. S. B. (1999). Guillotine Subdivisions Approximate Polygonal Subdivisions: A Simple Polynomial-Time Approximation Scheme for Geometric TSP, k-MST, and Related Problems. SIAM Journal on Computing, 28(4), 1298–1309. doi: 10.1137/s0097539796309764

3. Grigni, M., Koutsoupias, E., & Papadimitriou, C. (n.d.). An approximation scheme for planar graph TSP. Proceedings of IEEE 36th Annual Foundations of Computer Science. doi: 10.1109/sfcs.1995.492665

4. Klein, P. (n.d.). A linear-time approximation scheme for planar weighted TSP. 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS05). doi: 10.1109/sfcs.2005.7

5. Svensson, O. (2013). Overview of New Approaches for Approximating TSP. Graph-Theoretic Concepts in Computer Science Lecture Notes in Computer Science, 5–11. doi: 10.1007/978-3-642-45043-3_2

6. http://algo2.iti.kit.edu/schultes/lebenslauf/publications/euclTSPsummary.pdf

7. https://en.wikipedia.org/wiki/Travelling_salesman_problem

8. https://theory.epfl.ch/osven/courses/Approx13/Notes/lecture4-5.pdf

9. https://wushanshan.github.io/files/Algo_project.pdf

10. http://algo2.iti.kit.edu/schultes/lebenslauf/publications/euclTSPsummary.pdf