Survey Paper on Euclidean TSP

Zhengyu Li, University of Florida

***Abstract*—This survey paper introduces the Euclidean Travelling Salesman Problem (TSP) algorithm along with the NP, NP-hardness and NP-completeness proofs, and summarizes and classifies research results and methods on effectively solving Euclidean TSP problem and other NP-hard problem using the approximation and optimization algorithms.**

***Index Terms*—Approximation algorithm, Euclidean, greedy algorithm, optimization, TSP**

1. Introduction

The Euclidean TSP can be expressed as a salesman travels all cities in a given set V where V contains a list of cities. Depart from a city $i \in V$ to another city $j \in V$ until all cities in V were visited exactly one time and eventually return to the original city i. Another given set E contains all possible routes between cities in V. The distance is Euclidean distance on a d-dimensional space. Therefore, the problem can be shown in a graph $G=(V,E)$. What needs to be solved is the shortest distance route during the whole travel. The brute force algorithm on this problem, which is to check all possible situations and select the best case, would take exponential time. [6] There is no polynomial time algorithm to solve the exact optimal solution on this problem up to now. Therefore, the approximately method of this problem is important since the approximation algorithm could generally be able to give a much better computational time which is in polynomial time and give an acceptable solution on this problem. Besides that, some optimization algorithms such as genetic algorithm could also useful on this problem and even give both better solution and computational time. Some concepts involved in this survey paper are NP, NP-hardness and NP-complete. [1] NP is for decision problems, exists a solution that can verify the result in polynomial time. NP-hard is a problem P which can be reducible in polynomial time to some other known NP problem Q so that the $Q \leq P$ meaning the problem P is at least as hard as known NP-hard problem Q. NP-complete is a problem both in NP and NP hard. Therefore, NP-hard problem is "at least as hard as the hardest problems in NP" [2]. In order to solve Euclidean TSP and similar NP-hard problems faster, the polynomial time approximation algorithms and optimization algorithms are summarized and concluded in this survey paper.

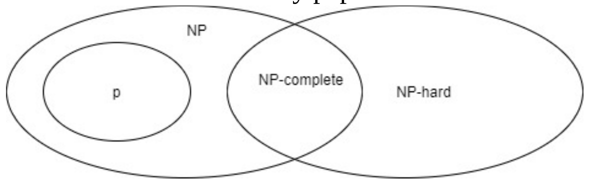

Fig. 1. The relationship between P, NP, NP-complete and NP-hard.

2. Overview of TSP NP-Completeness/Hard Proof

In 1975, C. Papadimitriou and R. Karp proved Euclidean TSP problem was a NP-complete problem. [3] It is a special case of metric TSP. For a NP-complete problem, it must be both NP and NP-hard problem. However, there also exists other version of TSP that was not in NP problem and only in NP-hard problem since the decision procedure was not the same. First of all, the prove of decision version of TSP when d = 2 is shown as below.

The input of a function contains three parameters which are G = (V, E) and k, where G is the graph contains all cities along with the cost of each existed path between every two cities in G. k is the maximum cost the salesman; thus, the totally cost of the travel should be less than or equal to k. In this case, to verify the solution is correct, the idea is to check each path $e \in E$ in the solution set that contains all cities in V and computes the summation of each cost less than the upper bound of the cost k which would take polynomial time. Therefore, verifying the solution takes polynomial time so that it is in NP problems.

Then, the input of another version of TSP does not have the cost k identified which means it is not a decision problem. Other procedures are exactly the same as the decision TSP except optimal solution verification. To check if the result of a solution is an optimal solution without giving minimal cost k would take more than polynomial time since the global shortest cost must be computed and it was proved more than polynomial time in 1972 by Dr. Karp. [5]

TSP could be proved as the NP-hard problem by using the reduction method. The Hamiltonian Cycle (HC), which was proved to be a NP-complete problem, could be used to prove Euclidean TSP is NP-hard problem by using the lemma 1. [4]

Lemma 1 *If L is a language $L' \leq_p L$ for some L' is an NP-complete problem, then L is NP-hard.*

Thus, a NP-complete problem Q $\leq_p$TSP need to be proved. An instance G = (V, E) of HC was found to be able to polynomial time reducible to TSP. [4] Suppose G' = (V, E′) is an instance of TSP where E′ contains all possible edges in the graph, the cost of E′ is 0 if $E' \in E$, otherwise cost of E' is 1. This reduction can be done in polynomial time and the correctness was also proved. [4] Therefore, TSP problem is one of NP-hard problems. It is significant to prove a problem as a NP-hard problem because the research efforts of different NP-hard problems can be combined together, and if one problem has a better solution, all other similar problems will have one. Exact cover problem can be used to prove Euclidean TSP problem in this way as well. [3]

3. Overview of Approximation Algorithm solution

Under certain conjectures of computational complexity theory such as such P≠NP conjecture, and efforts of previous researchers, the polynomial time solution is not existed for TSP. [6] Therefore, approximation algorithms are introduced in order to solve TSP approximate optimal solution in polynomial time. [7] There are many algorithms developed such as random and heuristic search algorithms including polynomial time approximation scheme (PTAS) tour construction methods, simulated annealing algorithm (SA), genetic algorithms (GA) and ant colony optimization algorithms.

3.1 PTAS Tour construction methods:

Tour construction method is a basic method using greedy algorithm to add an unvisited city in each round, until all cities are included. The first algorithm for the TSP problem in the plane d = 2 was provided by Arora in 1996. [17] The algorithm was using dynamic programming by recursively divide TSP to subproblems. First change the input points to unit grid points, then randomly divide problem into some sub-problems and using bottom up programming method to get the optimal solution. The running time is $n^{O(\frac{1}{\varepsilon})}$. The approximation ratio has an upper bound 2 of optimal solution which was also proved by Arora in 1998. [17] However, the algorithm sometimes failed and it was not easy to determine the failure. The solution was to use a derandomized version algorithm but the complexity would be much higher. Therefore, in 1998, Dr. Rao and Smith improved Arora’s method and provided both deterministic and randomized algorithms and obtained a better computational complexity which is O(*N*log*N*) and a better success probability which was 0.5. Beside these two significant methods, there are also many other methods in constructing tours such as nearest neighbor algorithm by keeping finding the nearest neighbor city takes O(n^2). The original algorithm which is to double minimal spanning tree (MST) can also achieve an approximately optimal solution with the worst-case approximation ratio 2 in O(n^2logn). The modification version of this algorithm by matching the one-degree node in MST can achieve a better approximation ratio to 1.5. []

3.2 Intelligent algorithms methods:

Besides traditional approximation algorithms, there are also some intelligent algorithms that can solve Euclidean TSP and the performance would be better than original methods. One algorithm is the simulated annealing algorithm. [9] With this method, the solution may not be the global optima since the gradient descent method is used to find the minima. However, the simulated annealing algorithm introduces random factors into the search process. The simulated annealing algorithm accepts a solution worse than the current one with a certain probability, so it may jump out of the local optimal solution and reach the global optimal solution. To compute the certain probability, the Metropolis algorithm is needed. The formulation of this algorithm (1) is shown below:

$$P = \begin{cases} 1 \ if \ E(x_{new} < x_{old}) \\ exp(-\frac{E(x_{new})-E(x_{old})}{T}) \ if \ x_{new} \geq x_{old} \end{cases} \quad (1)$$

It shows that if the solution is better after the moving, the algorithm could always accept the moving in order to find the global optima.

Algorithm 1

```
1  while(T > T_min):
2      dE = f(Y(i+1)) – f(Y(i));
3      if( dE >= 0)
4          Y(i+1) = Y(i);
5      else
6          if( exp( dE/T ) > random(0, 1))
7              Y(i+1) = Y(i);
8      T = r * T;
9      i++;
```

Pseudocode of Metropolis algorithm to find the global optima.

The Euclidean TSP can also be solved by another intelligent algorithm which is genetic algorithm. Genetic algorithm is inspired by natural selection and usually is able to give the high-quality solutions to search and optimization problems. [10] Every city in TSP can be seen as gene; every single route can be seen as the chromosome; population is a collection of chromosomes; parents can be seen as two cities which can create a route towards to an optimal solution; fitness is the metric to check how shortest this distance is; and mutation is to check some other routes which may give a better solution by replacing with current routes. [11]

The procedure of the genetic algorithm can be stated as the following: [17]

1. *Create a population of chromosomes*
2. *Compute the fitness of every chromosome*
3. *Select parents from the population in a certain ratio*
4. *Select two parents using crossover operator*
5. *Mutation procedure in order to optimize the result*
6. *Feed the factors to the next generation until the result does not change anymore or reach threshold.*

The genetic algorithm takes all cities and paths as the input and after selection, crossover and mutation procedures, the algorithm could get a first-generation solution and then carry the best result into the next generation. It stops until the fitness does not changing or it reaches the generation threshold. [12] The computational complexity of this algorithm generally is O(mn) with iteration number m and the input size n [13].

Ant Colony Algorithm, which is a bionic algorithm to simulate the behavior of ant colony, could also very useful on Euclidean TSP problem. Unlike the previous intelligent algorithms, ant colony algorithm is closer to heuristic construction algorithm more than optimization algorithms. Each ant completes the route alone, communicates through the pheromone which is a positive feedback, and eventually gathers to a locally optimal path. [14]

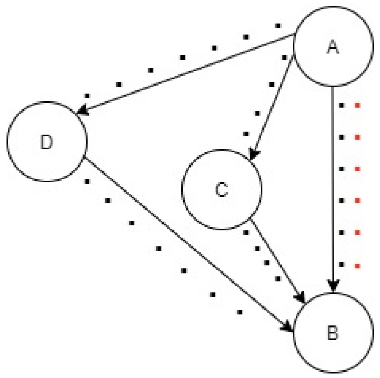

Fig. 2. The ant colony algorithm in TSP. A,B,C,D are four cities. Black points are pheromone and red points are feedback.

The algorithm procedure just like that the Fig.2 is shown as, ants start at city A and go to city B. There are three routes AB, ACB and ADB to go to B. At the beginning each path has one ant and they all leave pheromone in the same amount of distance. When the ant on path AB arrives at the city B after 6 units, other ants are still on their way. After 14 units, the ant on the farthest path ADB arrives B. However, the ant on the path AB already back to A. The ratio is almost 2:1. Therefore, in the next round, there are 2 ants in path AB and one ant in other paths. Eventually, ant colony would discard the other and select the shortest path AB.

Beside these algorithms, many researchers tried to seek and improve the performance of solving TSP by using the mixture of some algorithms. For example, an algorithm which combines both ant colony and association rule algorithm could be able to have a better solution which is closer to the exact optimal solution with the same computation complexity. [15]

4. Future Work

Euclidean TSP is a classic problem that first presented around 1930s and it is still an active and important research topic for many researchers. The future work on optimization of TSP problem includes solving other combinatorial optimization problems, computation complexity analysis for relatively new algorithms, verifying algorithms on other NP-hard problems and applying to new applications. Besides, many researchers show that integration of optimization algorithms would improve the performance so that newly integrated algorithms on solving TSP problem is a significant area to be figured out. Computation complexity analysis of some new algorithms like ant colony algorithm and robustness is an actively and rapidly growing topic as well. Moreover, TSP was proposed for transportation, such as airline arrangement, mail delivery etc. However, it has more other real-world applications in recent days such as multi-cooperative unmanned aerial vehicles mission planning method and automatic vehicle planning etc.

5. Conclusions

Euclidean TSP problem is one of typical problems in NP hard problems. Since all NP-hard problem can be transformed to 3SAT problem, the method could generally be applied to all other NP-hard problems which is potentially able to solve many real-world problems and create applications. Therefore, it is significantly to solve TSP. When solving TSP problems, since the exact solution would take more than polynomial time, the approximation algorithms and optimization algorithms are used to seek the almost best solution with a better computational complexity so that the TSP could be solved even if it becomes bigger and eventually including all places in the world or more. This paper shows Arora and Rao and Smith PTAS algorithms which can lower the running time to O(*N*log*N*) and three optimization algorithms. The approximation ratio and computational complexity could be relatively small and is becoming smaller as many new algorithms and methods are presented.

6. References

[1] Evan Klitzke 2017, “The Traveling Salesman Problem Is Not NP-complete”, https://eklitzke.org/the-traveling-salesman-problem-is-not-np-complete,

[2] Kleinberg & Tardos, Algorithm Design, 2005, ISBN 0132131080, 9780132131087

[3] Christos H. Papadimitriou, Richard Karp, 1975, The Euclidean traveling salesman problem is NP-complete

[4] C. L. R. S. 2009, Introduction to Algorithms 3rd edition, ISBN-13: 978-0262033848

[5] R.M. Karp, 1972, Reducibility among combinatorial problems, Complexity of Computer Computations, Plenum Press, New York

[6] Sanjeev Arora and Boaz Barak, 2013, Computational Complexity: A Modern Approach, ISBN-13: 978-7510042867

[7] Arora, S., Grigni, M., Karger, D., Klein, P., AND Woloszyn, 1998, A polynomial-time approximation scheme for weighted planar graph TSP. ACM, New York, pp. 33-41

[8] Z Xiang, Z Chen, X Gao, X Wang, F Di, L Li, G Liu, and Yi Zhang, 2015, Solving Large-Scale TSP Using a Fast Wedging Insertion Partitioning Approach, Mathematical Problems in Engineering

[9] Chi-Hwa Song, Kyunghee Lee and Won Don Lee, 2003, Extended Simulated Annealing for Augmented TSP and Multisalsemen TSP

[10] Introduction to Optimization with Genetic Algorithms, https://towardsdatascience.com/introduction-to-optimization-with-genetic-algorithm-2f5001d9964b, 2018

[11] Evolution of a salesman: A complete genetic algorithm tutorial for Python, https://towardsdatascience.com/evolution-of-a-salesman-a-complete-genetic-algorithm-tutorial-for-python-6fe5d2b3ca35, 2018

[12] Jan Scholz, 2018, Genetic Algorithms and The Travelling Salesman Problem A Historical Review

[13] Asher Saban, Liat Waltuch, 2012, Genetic Algorithm for the Traveling Salesman Problem

[14] Hassan Ismkhan, Kamran Zamanifar, 2014, Using Ants as a Genetic Crossover Operator in GLS to Solve STSP,

[15] S. Gao, L. Zhang, F. Zhuang, C. Zhang, 2007, Solving Traveling Salesman Problem by Ant Colony Optimization Algorithm with Association Rule

[16] Abid Hussain, Yousaf Shad Muhammad, 2017, Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator

[17] Arora, 1998, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems

[18] Satish B. Rao & Warren D. Smith, 1998, Improved approximation schemes for geometrical graphs via “spanners" and “banyans"

[19] Christian Nilsson, 2003, Heuristics for the Traveling Salesman Problem