The Travelling Salesman Problem, Applications and Solvers

Laurik Helshani
PhD candidate at European University of Tirana
helshani@gmail.com

Ervin Ramollari
Lecturer at University of New York Tirana
ervinramollari@unyt.edu.al

Abstract

The travelling salesman problem (TSP) has puzzled mathematicians and computer scientists for many decades and it has found applicability in a number of diverse domains. The aim of this paper is to present the theory of TSP and its classification into the symmetric travelling salesman problem (sTSP), asymmetric travelling salesman problem (aTSP) and multi travelling salesman problem (mTSP), as well as the different mathematical formulations of the problem. This paper also provides an overview of the applications of TSP, and in the end presents the various traditional and modern approaches that have been proposed to solve this important problem.

Keywords: *Travelling Salesman Problem (TSP), Graph Theory, exact and non exact solver*

Introduction

The travelling salesman problem (TSP) has been proved to be a NP-hard problem, i.e. it is not possible to obtain a polynomial time algorithm to obtain an optimal solution. TSP is easy to interpret, yet hard to solve. This problem has aroused many scholars' interests since it was put forward in 1932. However, so far, no effective solution has been found.

Although TSP only represents a problem of the shortest path, in actual life, many physical problems are found to be the TSP (Yuan-bin, 2011).

Origin

TSP was studied in the 18th century by a mathematician from Ireland named Sir William Rowam Hamilton and by the British mathematician named Thomas Penyngton Kirkman. Detailed discussion about the work of Hamilton & Kirkman can be found in the book titled "Graph Theory" (Biggs, Lloyd, & Wilson, 1976). It is believed that the general form of the TSP has been first studied by Karl Menger in Vienna and Harvard.

Definition

In the traveling salesman problem (TSP) a travelling salesman needs to promote products in n cities (including the city where he lives). After visiting each city (each city can be visited only once), he returns to the departure city. Let us suppose that there exists a road connecting every two cities. What is the best route to follow in order to minimize the distance of the journey? (Yuan-bin, 2011)

Complexity

Given n is the number of cities to be visited, the total number of possible routes covering all cities can be given as a set of feasible solutions to the TSP and is given by (n-1)!/2.

Classification

The TSP is broadly classified into symmetric travelling salesman problem (sTSP), asymmetric travelling salesman problem (aTSP) and multi travelling salesman problem (mTSP).

sTSP

Let $V = \{v_1, \ldots, v_n\}$ be a set of nodes (cities), $A = \{(r,s) : r,s \in V\}$ be the set of edges connecting nodes, and $d_{rs} = d_{sr}$ be a cost measure associated with edge $(r,s) \in A$.

The sTSP is the problem of finding a minimal length closed tour that visits each node exactly once. In this case nodes $v_i \in V$ are given by their coordinates (x_i, y_i) and d_{rs} is the Euclidean distance between r and s then we have an Euclidean TSP.

aTSP
If $d_{rs} \neq d_{sr}$ for at least one (r, s) then the TSP becomes an aTSP.

mTSP
The mTSP is defined as follows. In a given set of nodes, let there be m salesmen located at a single depot node. The remaining nodes (cities) that are to be visited are intermediate nodes. Then, the mTSP consists of finding tours for all m salesmen, who all start and end at the depot, such that each intermediate node is visited exactly once and the total cost of visiting all nodes is minimized (Matai & Singh, 2010).

Mathematical formulations of TSP

Mathematical formulation as graph

The TSP can be defined on a complete undirected graph G = (V, E) if it is symmetric or on a directed graph G = (V, A) if it is asymmetric. The set V ={1,..., n} is a set of nodes, E = {(i, j) : i, jV, i j} is a set of edges and A = {(i, j) : i, jV, i < j} is a set of arcs (directed). A cost matrix C = (c_{ij}) is defined on E or on A. The cost matrix satisfies the triangle inequality whenever $c_{ij} \leq c_{ik} + c_{kj,}$ for all i, j, k. In particular, this is the case of planar problems for which the vertices are points Pi = (Xi,Yi) in the plane, and $c_{ij} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$ is the Euclidean distance. The triangle inequality is also satisfied if cij is the length of a shortest path from I to j on G. (Matai & Singh, 2010). The following table links the concepts in TSP with corresponding concepts in graph theory.

TABLE 1 – Link between TSP and Graph Theory

Travelling Salesman Problem	Graph Theory
City	Node (or Vertex)
Connection between cities	Edge
Distance	Edge's length
Set of cities and connections	G = (V, E); Graph = (Set of nodes, Set of edges)
Structure consisting of cities and connections (of given distances) undirected	Weighted and undirected graph
If all pairs of cities are connected	Complete graph
Trip	Subset of graph edges
Tour	Hamiltonian cycle
Travelling salesman problem	Search of Hamiltonian cycle in the complete graph

Linear programming formulation

According to the definition of the TSP, its mathematical description is as follows:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$
$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for } i = 1 \dots n \tag{2}$$
$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for } j = 1 \dots n \tag{3}$$
$$\sum_{\{i,j\}\subseteq S} x_{ij} \leq |S| - 1 \ \forall\, S \subset V,\ |V| \geq 2 \tag{4}$$
$$x_{ij} \in \{0, 1\}\ i, j = 1,2, \in n\ i \neq j \tag{5}$$

Where d_{ij} means the distance between the city i and city j; decision variable = 1 means the route the salesman passes through (including the route from city i and city j); = 0 means the route which isn't chosen by the salesman.

Objective function (1) means the minimum total distance; (2) means that a salesman can departure from city i only once; (3) means that a salesman can enter city j only once; (2) and (3) serve to ensure that the salesman visits each city once, but it doesn't rule out the possibility of any loop; (4) requires that no loop in any city subset should be formed by the salesman; |S| denotes the number of elements included in the set S. (Kämpf, 2006)

Applications of TSP

Computer wiring

(Lenstra & Kan, 1974) reported a special case of connecting components on a computer board. Modules are located on a computer board and a given subset of pins has to be connected. In contrast to the usual case where a Steiner tree connection is desired, here the requirement is that no more than two wires are attached to each pin. Hence we have the problem of finding a shortest Hamiltonian path with unspecified starting and terminating points. A similar situation occurs for the so-called test bus wiring. To test the manufactured board one has to realize a connection which enters the board at some specified point, runs through all the modules, and terminates at some specified point. For each module we also have a specified entering and leaving point for this test wiring. This problem also amounts to solving a Hamiltonian path problem with the difference that the distances are not symmetric and that starting and terminating point are specified.

School bus routing problem

(Angel, Caudle, Noonan, & Whinston, 1972) investigated the problem of scheduling buses as a variation of the mTSP with some side constraints. The objective of the scheduling is to obtain a bus loading pattern such that the number of routes is minimized, the total distance travelled by all buses is kept at minimum, no bus is overloaded and the time required to traverse any route does not exceed a maximum allowed policy.

Vehicle routing

Suppose that in a city n mail boxes have to be emptied every day within a certain period of time, say 1 hour. The problem is to find the minimum number of trucks to do this and the shortest time to do the collections using this number of trucks. As another example, suppose that n customers require certain amounts of some commodities and a supplier has to satisfy all demands with a fleet of trucks. The problem is to find an assignment of customers to the trucks and a delivery schedule for each truck so that the capacity of each truck is not exceeded and the total travel distance is minimized. Several variations of these two problems, where time and capacity constraints are combined, are common in many real world applications. This problem is solvable as a TSP if there are no time and capacity constraints and if the number of trucks is fixed. In this case we obtain an m - salesmen problem. Nevertheless, one may apply methods for the TSP to find good feasible solutions for this problem. (Lenstra & Kan, 1974)

Design of global navigation satellite system surveying networks

A very recent and an interesting application of the mTSP, as investigated by (Saleh & Chelouah, 2004), arises in the design of global navigation satellite system (GNSS) surveying networks. A GNSS is a space-based satellite system which provides coverage for all locations worldwide and is quite crucial in real-life applications such as early warning and management for disasters, environment and agriculture monitoring, etc. The goal of surveying is to determine the geographical positions of unknown points on and above the earth using satellite equipment. These points, on which receivers are placed, are co-ordinated by a series of observation sessions. When there are multiple receivers or multiple working periods, the problem of finding the best order of sessions for the receivers can be formulated as an mTSP.

TSP solvers

Exact Solvers

There are two groups of exact solvers. One of these is solving relaxations of the TSP Linear Programming formulation and uses methods like Cutting Plane, Interior Point, Branch-and-Bound and Branch-and-Cut. Another smaller group is using Dynamic Programming. For both groups the main characteristic is a guarantee of finding optimal solutions at the expense of running time and space requirements. (Chauhan, Gupta, & Pathak, 2012)

Branch and Bound
The Branch and Bound method implicitly enumerates all the feasible solutions, using calculations where the integer constraints of the problems are relaxed. In other words the branch and bound strategy divides a problem to be solved into a number of sub-problems. It is a system for solving a sequence of sub-problems each of which may have multiple possible solutions and where the solution chosen for one sub-problem may affect the possible solutions of later sub-problems. To avoid the complete calculation of all partial trees, we first try to find a practical solution and note its value as an upper bound for the optimum. (Chauhan et al., 2012)(Chauhan et al., 2012)

Branch and cut
The branch and cut method solves the linear program without the integer constraint using the regular simplex algorithm. When an optimal solution is obtained, and this solution has a non-integer value for a variable that is supposed to be integer, a cutting plane algorithm is used to find additional linear constraints which are satisfied by all feasible integer points but violated by the current fractional solution. If such an inequality is found, it is added to the formulation, such that resolving it will yield a different solution which is hopefully "less fractional". This process is repeated until either an integer solution is found (which is then known to be optimal) or until no more cutting planes are found. We may normally end with an optimal solution however, in practice we may not have an exact separation algorithm and it may return no violated inequality although there are some. If we have not terminated with an optimal solution to IP, we branch. (Chauhan et al., 2012)

Dynamic Programming
It is a technique for efficiently computing recurrences by storing partial results and re-using them when needed. It is well known that dynamic-programming

recursions can be expressed as shortest-path problems in a layered network whose nodes correspond to the states of the dynamic program. (Chauhan et al., 2012)

Non-Exact Solvers

These solvers offer potentially non-optimal but typically faster solutions. In a way they are the opposite trade-off of the exact solvers. Non-exact solvers can be subdivided into:

Approximation Algorithms
These algorithms come with a worst case approximation factor for the found solution.

Heuristic Algorithms
These algorithms only promise a feasible solution. They range from simple tour-construction methods like Nearest Neighbor. Here we find methods like Simulated Annealing, Genetic Algorithms, Ant Colony Algorithms and machine learning algorithms like Neural Networks. (Chauhan et al., 2012)

Simulated Annealing Algorithm (SA)
Simulated Annealing is inspired by the process of annealing in metallurgy. In this natural process a material is heated and slowly cooled under controlled conditions to increase the size of the crystals in the material and reduce their defects. This has the effect of improving the strength and durability of the material. The heat increases the energy of the atoms allowing them to move freely, and the slow cooling schedule allows a new low-energy configuration to be discovered and exploited. (Brownlee, 2011)
Solution of TSP using Simulated Annealing Algorithm
In simulated annealing algorithm, the problem of finding an optimal solution is analogous to a process of reducing the overall "energy" of a system via state transitions. In the traveling salesman problem, a state is defined as a route, or a permutation of the cities to be visited. The energy of a state is the total distance of a route. In every iteration of the algorithm, a candidate state transition, also known as a neighbor of the current state, is generated by exchanging a pair of cities. (Yang, 2010)
Genetic algorithms
As a computational intelligence method is a search technique used in computer science to find approximate solutions to combinatorial optimization problems. The genetic algorithm are more appropriately said to be an optimization technique based on natural evolution. Like evolution, these algorithms are

based on the idea of the survival of the fittest. (Dwivedi, Chauhan, Saxena, & Agrawal, 2012) Genetic algorithms treat candidate solutions as members of a population, in which the fittest members (best solutions) are the ones that are selected and allowed to reproduce to produce children (child solutions). In these algorithms three main operators: reproduction, crossover and mutation, are applied on the population to create a new population.

Solution of TSP using Genetic Algorithm

First, create a group of many random tours in what is called a population. This algorithm uses a greedy initial population that gives preference to linking cities that are close to each other. Second, pick 2 of the better (shorter) tours parents in the population and combine them to make 2 new child tours. Hopefully, these children tour will be. In a small percentage of the time, the child tours are mutated. This is done to prevent all tours in the population from looking identical. The new child tours are inserted into the population replacing two of the longer tours. The size of the population remains the same. New children tours are repeatedly created until the desired goal is reached. (Vishnupriyan, Govindarajan, Prabhakaran, & Ramachandran, 2008)

Conclusions

In this paper we presented the theory of the travelling salesman problem (TSP), classification, and its mathematical formulations. Here we provided an overview of some interesting application areas where this problem appears and where an efficient solution would be beneficial. As TSP is a NP-hard problem and hard to solve, thus we presented a survey of the different solutions that have been proposed to date to this problem. They have been classified into exact and non-exact solvers. Exact solvers guarantee of finding optimal solutions at the expense of running time and space requirements. On the other hand, non-exact solvers, such as Genetic Algorithms and Simulated Annealing, offer potentially non-optimal, but typically faster solutions. For sufficintly large numbers of graph nodes, non-exact solvers can be the only feasible choice, due to time and space constraints.

References

Angel, R. D., Caudle, W. L., Noonan, R., & Whinston, A. (1972). Computer-assisted school bus scheduling. Management Science, 18(6), B–279.

Biggs, N., Lloyd, E. K., & Wilson, R. J. (1976). Graph Theory, 1736-1936. Oxford University Press.

Brownlee, J. (2011). Clever algorithms: nature-inspired programming recipes. Jason Brownlee.

Chauhan, C., Gupta, R., & Pathak, K. (2012). Survey of methods of solving TSP along with its implementation using dynamic programming approach. International Journal of Computer Applications, 52(4).

Dwivedi, V., Chauhan, T., Saxena, S., & Agrawal, P. (2012). Travelling Salesman Problem using Genetic Algorithm. IJCA Proceedings on Development of Reliable Information Systems, Techniques and Related Issues (DRISTI 2012), (1), 25.

Kämpf, M. (2006). Probleme der Tourenbildung. TU, Fak. für Informatik. Retrieved from http://monarch.qucosa.de/fileadmin/data/qucosa/documents/5310/data/CSR-06-04.pdf

Lenstra, J. K., & Kan, A. R. (1974). Some simple applications of the travelling salesman problem. JSTOR. Retrieved from http://www.jstor.org/stable/pdf/3008306.pdf

Matai, R., & Singh, S. P. (2010). Traveling Salesman Problem: An Overview of Applications, Formulations, and Solution Approaches. Tech. Www. Intechopen. Com.

Saleh, H. A., & Chelouah, R. (2004). The design of the global navigation satellite system surveying networks using genetic algorithms. Engineering Applications of Artificial Intelligence, 17(1), 111–122.

Vishnupriyan, S., Govindarajan, L., Prabhakaran, G., & Ramachandran, K. P. (2008). Quality Improvement in Higher Education through Normalization of Student Feedback data Using Evolutionary Algorithm. International Journal of Applied Management and Technology, 6(3).

Yang, F. (2010). Solving Traveling Salesman Problem Using Parallel Genetic Algorithm and Simulated Annealing. Massachusetts Institute of Technology. Retrieved from http://courses.csail.mit.edu/18.337/2010/projects/reports/Fan_Yang_Final_Project_Report_Fixed_Figure_References.pdf

Yuan-bin, M. (2011). The Advantage of Intelligent Algorithms for TSP. Retrieved from http://repositories.vnu.edu.vn/jspui/handle/123456789/16962