TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 15, No. 2, August 2015, pp. 390 ~ 396
DOI: 10.11591/telkomnika.v15i2.8301

Heuristic Approaches to Solve Traveling Salesman Problem

Malik Muneeb Abid[1]*, Iqbal Muhammad[2]
[1]School of Transportation and Logistics, Southwest JiaoTong University, Chengdu, China, 610031
[2]School of Information Sciences and Technology, Southwest JiaoTong University, Chengdu, China.
*Corresponidng author, e-mail: malik.muneeb.abid@hotmail.com

Abstract

This paper provides the survey of the heuristics solution approaches for the traveling salesman problem (TSP). TSP is easy to understand, however, it is very difficult to solve. Due to complexity involved with exact solution approaches it is hard to solve TSP within feasible time. That's why different heuristics are generally applied to solve TSP. Heuristics to solve TSP are presented here with detailed algorithms. At the end, comparison among selected approaches shows the efficiency of approaches in terms of solution quality and time consumed to solve TSP.

Keywords*: TSP, heuristics, survey, NP-hard*

1. Introduction

The computational complexity theory makes it possible to validate the concepts of "easy" and "hard" problems and the distinction among them. Problems can be formally classified based on their complexity and if a problem strictly belongs to the family of NP-hard or complete problems, we know in advance that there is little chance of finding an efficient and exact algorithm to solve it. The algorithm for such a problem has an execution time bursting for increasing problem sizes, and in majority cases is not feasible for most practical purposes. Computers are playing every effective role in solving different complex problems but the matter of fact is that some problems are fundamentally harder to solve. Although, for some problems it is possible to develop intelligent algorithms that solve the problems expeditiously, however, it seems substantially hard even in some cases impossible to come up with any solution [24].

Davendra [13] defined TSP as, "Given a set of cities of different distances away from each other, and the objective of TSP is to find the shortest path for a salesperson to visit every city exactly once and return back to the origin city". TSP is an important applied problem with many fascinating variants; like theoretical mathematics, computer science, NP hard problem, combinatorial optimization and operation research [25]. TSP is classified as symmetric, asymmetric and multiple TSP based on the distance and direction between two cities in a graph (Figure 1). If distance between two cities is same in each direction it is symmetric with undirected nature otherwise it is asymmetric.

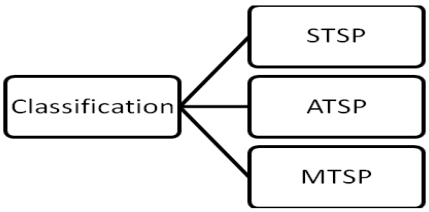

Figure 1. Classification of TSP

TSP can be solved by using optimal algorithms (e.g., IP formulation), however, these solutions are computationally infeasible and it is almost impossible to generate optimal solution

Received April 23, 2015; Revised July 5 2015; Accepted July 20, 2015

within reasonable time. For this reason heuristics are used to solve TSP. This paper provides a survey of different approaches used to solve the TSP heuristically.

1.1. Illustration of TSP

The core objective of TSP is to minimize the total traveling cost of object around tours. In order to understand TSP, let us explore the given below example. Figure 2 shows the road distance between the three towns i.e. ABC and additionally assume that a salesperson, whose business is to sells lubricant items to different companies located in these three cities, must travel (starting from home). Here the decimal values near the line edges in the diagram are the driving distances between the cities. In this example, we are assuming that we have a symmetric TSP i.e. the cost in going from A to B is the same as the cost in going from B to A but in asymmetric TSP the cost could not be the same between the two points.

In the given situation, one question arises in mind that how many TSP tours could be? The general answer to this question is, for the complete graph with n vertices, the number of different TSP routes would be Equation (1).

$$T = \frac{(n-1)!}{2} \quad (1)$$

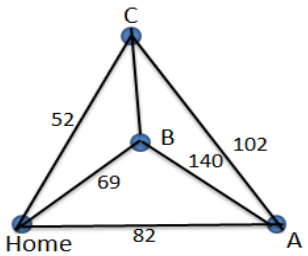

Figure 2. A 4 city TSP problem

Let us calculate the cheapest tour by working on above Figure 2.

HABCH→82+140+90+52=364
HACBH→82+102+90+69=343
HCABH→52+102+140+69=363

Thus, going from H to A, then to C, then to B and then back H could be the best choice.

1.2. Related Work

In literature TSP is used in two forms: i) combinatorial optimization version and ii) decision version. In first version it is used to find a minimum Hamiltonian cycle and in later version to check the existence of smaller graph.

Theoretical computer science and operations research, both fields of combinatorial optimization contains TSP. In this problem, set of cities are given with their distances to find the shortest route to each city without visiting a city twice. In 1930, it was first formulated as a mathematical model and applied to so many areas to find their optimal solutions e.g., Clustering of array of data [1], Handling of a warehouse materials [2] and crystal structure analysis [3]. Resource constrained scheduling problem with aggregate deadline also solved with TSP [4]. Researches [5, 6] took orienteering and prize collection problems as special cases of resource constrained TSP. One of the best known and more complex combinatorial problem is Vehicle routing problem, to determine the order of vehicle for customers serving from fleet of vehicles, is being solved by TSP [7].

TSP is easy to understand but it's really difficult to solve it. In 1972, Richard M. Karp showed that it is NP-complete to solve Hamiltonian cycle problem, which explains the NP-hardness of TSP. This can be judged that how much computational difficulty is attached to find the optimal route [9]. With the passage of time TSP is getting more and more sophisticated and solving instances are larger now. A brief view of TSP milestones is given below in Table 1.

Table 1. Summary of the Milestones achieved in TSP (12)

Year	Research Team	Size of Instance
1954	G. Dantzig, R. Fulkerson, and S. Johnson [14]	49 cities
1971	M. Held and R.M. Karp [15]	64 cities
1975	P.M. Camerini, L. Fratta, and F. Maffioli [16]	67 cities
1977	M. Grötschel [17]	120 cities
1980	H. Crowder and M.W. Padberg [18]	318 cities
1987	M. Padberg and G. Rinaldi [19]	532 cities
1987	M. Grötschel and O. Holland [20]	666 cities
1987	M. Padberg and G. Rinaldi [19]	2,392 cities
1994	D. Applegate, R. Bixby, V. Chvátal, and W. Cook [21]	7,397 cities
1998	D. Applegate, R. Bixby, V. Chvátal, and W. Cook [22]	13,509 cities
2001	D. Applegate, R. Bixby, V. Chvátal, and W. Cook [23]	15,112 cities
2006	D. Applegate, R. Bixby, V. Chvátal, W. Cook, and K. Helsgaun [23]	24,978 cities

This paper is organized as follows: Next section provides the algorithm details of selected heuristic solutions for TSP. After that all of these techniques are analyzed and compared for solution quality and time consumed for different problem data. At last conclusions and future directions are presented.

2. Heuristic Solution Techniques

There are various heuristics and approximate solution approaches, which had been devised during last decades, to find solution within reasonable time and with 2-3 % optimality gap [8]. There are numerous approaches used to solve TSP shown in literature. Some important and mostly used approaches are enlisted here with their application algorithms. Variables for these algorithms are illustrated in Table 2.

Table 2. Illustration of Variables

Description	Variable
Initial solution provided by the user	S
Objective function value	z
The best solution	S*
Cities in TSP (Nodes in transportation network)	i, j
Population	P
Generations	G
Generation counter	N_{gen}
Initial temperature	T
Cooling factor	r
Number of times the temperature *T* is decreased	*ITEMP*
Maximum number of new solutions to be accepted at each temperature	*NLIMIT*
Maximum number of solutions evaluated at each temperature	*NOVER*
Gain parameter	*L*

2.1. Brute Force Method

Algorithm for TSP using Brute-force method contains the following steps:

Algorithm 1: TSP using Brute Force Method

Step 1: calculate the total number of tours (where cities represent the number of nodes).
Step 2: draw and list all the possible tours.
Step 3: calculate the distance of each tour.
Step 4: choose the shortest tour; this is the optimal solution.

2.2. Greedy Approach

Greedy approach solves TSP by using the five steps, given in Algorithm 2.

Algorithm 2: TSP using Greedy Approach

Step1: Look at all the arcs with minimum distance.
Step 2: Choose the *n* cheapest arcs
Step 3: List the distance of arcs starting from the minimum distance to maximum distance.
Step 4: Draw and check if it forms a Hamiltonian cycle.
Step 5: If step 4 forms a Hamiltonian cycle than we have an optimal solution; write down the tour of the optimal solution and calculate their distance.

2.3. Nearest Neighbor Heuristic

Algorithm 3 shows the methodology to solve TSP by Nearest Neighbor Heuristic.

Algorithm 3: TSP using Nearest Neighbor Heuristic

Step 1: Pick any starting node.
Step 2: Look at all the arcs coming out of the starting node that have not been visited and choose the next closest node.
Step 3: Repeat the process until all the nodes have been visited at least once.
Step 4: Check and see if all nodes are visited. If so return to the starting point which gives us a tour.
Step 5: Draw and write down the tour, and calculate the distance of the tour.

2.4. Branch and Bound Method

Branch and bound technique is commonly used optimization technique. Formulation of TSP by using branch and bound technique is given in Algorithm 4.

Algorithm 4: TSP using Branch and Bound Method

Step 1: Choose a start node.
Step 2: Set bound to a very large value, let's say infinity.
Step 3: Choose the cheapest arc between the current and unvisited node and add the distance to the current distance and repeat while the current distance is less than the bound.
Step 4: If current distance is less than bound, then we are done
Step 5: Add up the distance and bound will be equal to the current distance.
Step 6: Repeat step 5 until all the arcs have been covered.

2.5. 2-Opt Algorithm

For n nodes in the TSP problem, the 2-Opt algorithm consists of the steps shown in Algorithm 5.

Algorithm 5: TSP using 2-Opt Algorithm

Step 1:
Let S be the initial solution provided by the user and z its objective function value. Set S*=s, z*=z, i=1 and j=i+1=2.
Step 2:
Consider the exchange results in a solution S that has objective function value z'<z*, set z*=z' and S*=S'. If j<n repeat step 2; otherwise set i=i+1 and j=i+1. If i<n, repeat step 2; otherwise go to step 3.
Step 3:
If S≠S*, set S=S*, z=z*, i=1, j=i+1=2 and go to step 2. Otherwise, output S* as the best solution and terminate the process.

2.6. Greedy 2-Opt Algorithm

The greedy 2-Opt algorithm is a variant of 2-opt algorithm. It consists of the steps shown in Algorithm 6.

Algorithm 6: TSP using Greedy 2-Opt Algorithm

Step 1:
Let S be the initial solution provided by the user and z its objective function value. Set S*=s, z*=z, i=1 and j=i+1=2.

Step 2:
Transpose Node i and Node j, i<j. Compare the resulting objective function value z with z*. If z ≥z* and j<n, set j=j+1 and repeat step 2; Otherwise go to step 3.
Step 3:
If z<z*, set S*=S, z*=z, i=1, j=i+1=2 and go to step 2. If z ≥ z* and
j=n, set i=i+1, j=j+1 and repeat step 2. Otherwise, output S* as the best solution and terminate the process.

2.7. Genetic Algorithm

This algorithm is based on genetics. The Genetic Algorithm works as shown in Algorithm 7.

Algorithm 7: TSP using Genetic Algorithm

Step 0:
Obtain the maximum number of individuals in the population P and the maximum number of generations G from the user, generate P solutions for the first generation's population randomly, and represent each solution as a string. Set generation counter N_{gen}=1.
Step 1:
Determine the fitness of each solution in the current generation's population and record the string that has the best fitness.
Step 2:
Generate solutions for the next generation's population as follows:
1. Retain 0.1P of the solutions with the best fitness in the previous population.
2. Generate 0.89P solutions via mating, and
3. Select 0.01P solutions from the previous population randomly and mutate them.
Step 3:
Update N_{gen}= N_{gen}+1. If N_{gen} ≤ G, go to Step 1. Otherwise stop.

2.8. Simulated Annealing (SA)

Statistical mechanics is the basic idea of simulated annealing (SA). Analogy of the behavior of physical systems in the heat bath is main motivation of SA. Solution state is represented by the temperature. Initiating with an initial temperature, algorithm moves to the next temperature until it reaches a frozen state.

Algorithm 8: TSP using Simulated Annealing

Step 0: Set *S* = initial feasible solution
Step 1: Repeat step 2 *NOVER* times or until the number of successful new solutions is equal to *NLIMIT*
Step 2: Pick a pair of machines randomly and exchange their positions.
Step 3: Set *T* = *rT* and *ITEMP* = *ITEMP* + *1*. If *ITEMP* <= 100, go to step 1; otherwise stop.

2.9. Neural Network

In this process at each processing step a city is surveyed. On complete iteration visits all cities in a preset order. First of all a node is selected from randomly from the network and a gain parameter is used to attain the result and quality if solution. Algorithm 9 shows the working mechanism of neural network.

Algorithm 9: TSP using neural network

Step 1. Find the node j_c which is closed to city i.
Step 2. Move node j_c and its neighbors on the ring towards city i.
The distance each node will move is determined by a function f (L, n), where n is the distance measured along the loop between nodes j and j_c.
n =inf (j- j_c(mod N), j_c-j(mod N))
Every node j is moved from current position to a new one.
The function f is defined to be
$f(L,n)=sqrt(1/2)*exp(-n^2/G^2)$

3. Discussions

Previous section demonstrates the different methodologies to solve NP-hard TSP problem approximately. Although, different techniques had been devised previously, but, all available techniques are not efficient in terms of solution time and quality. Comparison by Maredia [10] shows that Nearest Neighbor heuristic works well but it is not sure that it will give us solutions good as brute force. Moreover, greedy approach is not a good approximation technique for TSP. Comparison of Branch and bound technique with brute force method is presented in Figure 3 (data extracted from Maredia [10]). SA algorithm obtains has ability to find the good quality final solutions by mechanism of gradually going from one solution to the next. The main difference of SA from the 2-opt is that the local optimization algorithm is often restrict their search for the optimal solution in a downhill direction which mean that the initial solution is changed only if it results in a decrease in the objective function value**.** However, it is shown by Kim et al. [11] that 2-opt algorithm works well when the problem size is less than 50 cities. For comparison of the approaches we are referring to data Kim et al. [11]. Data extracted from Kim et al. [11] is plotted in Figure 4 and 5, according to results it is obvious that the solution of TSP using the neural network outperforms than all other approaches for all problem sizes. However, if we compare the time consumed to solve the problems, it is easy to realize that neural network approach is taking much more time as compared to others. Genetic algorithm has been used for many optimization problems; however, the use of genetic algorithm for TSP has disadvantages of premature convergence and poor local search capability. These disadvantages can be overcome by adaptation of other efficient working algorithms e.g., artificial immune systems [13].

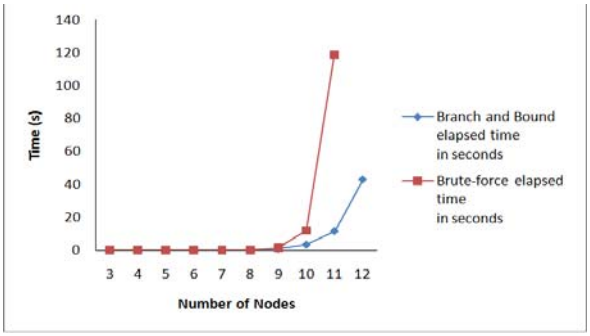

Figure 3. Comparison of Branch and Bound Technique with Brute Force Method

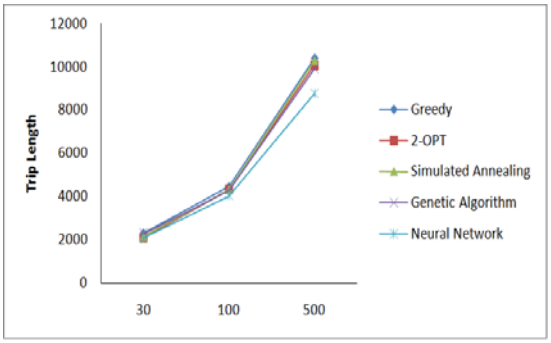

Figure 4. Length Comparison of TSP Heuristics

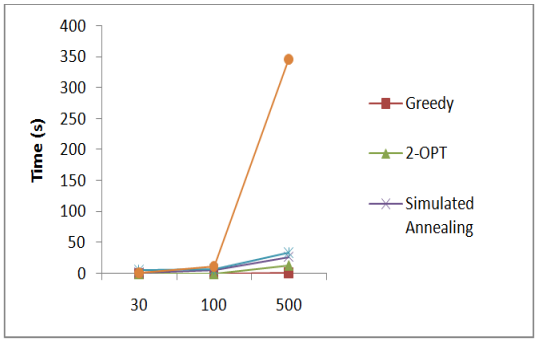

Figure 5. Time Comparison of TSP Heuristics

Advantages of these heuristics approaches are that they are simple and easy to understand. These require less programming and storage requirements and produce multiple solutions with in less time [26]. However, heuristics local improvements in heuristics can be source of lack in global prospective of objective function.

4. Conclusion and Recommendations

This paper provides the survey of different heuristics used to solve TSP. First, an example of TSP is presented to give the idea of TSP. This survey is limited to some selected approaches because it is not possible to cover all approaches here. So, only most relevant approaches are discussed here. Although a number of heuristics had been devised however; some are efficient with respect to time and some are outperforming in solution quality. Future work will consider the formulation of heuristics to solve the TSP in a reasonable time with bench mark problem data of mile stones presented in Table 1.

References

[1] Lawler EL, Lenstra JK, Rinnooy AHG, Shmoys DB. The Traveling Salesman Problem. Chichester: John Wiley. 1985.
[2] Ratliff HD, Rosenthal AS. *Order-Picking in a Rectangular Warehouse: A Solvable Case for the Traveling Salesman Problem*. PDRC Report Series No. 81-10. 1981.
[3] Bland RE, Shallcross DF. *Large Traveling Salesman Problem Arising from Experiments in X-ray Crystallography: a Preliminary Report on Computation.* Technical Report No. 730. 1987.
[4] Miller, Pekny J. Exact Solution of Large Asymmetric Traveling Salesman Problems. *Science 251.* 1991: 754-761.
[5] Balas E. The Prize Collecting Traveling Salesman Problem. *Networks 19.* 1989: 621-636.
[6] Golden BL, Levy L, Vohra R. The Orienteering Problem. *Naval Research Logistics 34.* 1987: 307-318.
[7] Christofides N. Vehicle Routing, in The Traveling Salesman Problem. In: Lawler, Lenstra, Rinooy Kan and Shmoys. *Editors*. John Wiley. 1985: 431-448.
[8] Rego C, Gamboa D, Glover F, Osterman C. Traveling salesman problem heuristics: leading methods, implementations and latest advances. *European Journal of Operational Research*. 2011; 211(3): 427-441.
[9] Cook W. History of the TSP. The Traveling Salesman Problem. Georgia Tech. 2009.
[10] Maredia A. History, Analysis, and Implementation of Traveling Salesman Problem (TSP) and Related Problems. Thesis. Department of Computer and Mathematical Sciences University of Houston-Downtown. 2010.
[11] Kim IB, Shim IJ, Zhang M. Comparison of TSP Algorithms, Project for Models in Facilities Planning and Materials Handling. 1998.
[12] http://www.math.uwaterloo.ca/tsp/history/milestone.html (assessed on 08-06-2015).
[13] Davendra D. Traveling Salesman Problem, Theory and Applications. INTECH open access publishers. 2010.
[14] Dantzig G, Fulkerson R, Johnson S. Solution of a large-scale traveling-salesman problem. *Operations Research 2.* 1954: 393-410.
[15] Held M, Karp RM. The traveling-salesman problem and minimum spanning trees: part II. *Mathematical Programming 1.* 1971: 6-25.
[16] Camerini PM, Fratta L, Maffioli F. On improving relaxation methods by modified gradient techniques. *Mathematical Programming Study 3.* 1975: 26-34.
[17] Grötschel M, Padberg M. On the symmetric traveling salesman problem: theory and computation. In: R. Henn, B. Korte, W Oettli. *Editors*. Lecture Notes in Economics and Mathematical Systems 157. Springer, Berlin. 1977: 105-115.
[18] Crowder H, Padberg MW. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science 26.* 1980: 495-509.
[19] Padberg M, Rinaldi G. Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters 6.* 1987: 1-7.
[20] Grötschel M, Holland O. A cutting plane algorithm for minimum perfect 2-matchings. *Computing 39.* 1987: 327-344.
[21] Applegate D, Bixby R, Chvátal V, Cook W. Finding cuts in the TSP (A preliminary report). *DIMACS Technical Report.* 1995: 95-05.
[22] Applegate D, Bixby R, Cook W, Chvátal V. On the solution of traveling salesman problems. Rheinische Friedrich-Wilhelms-Universität Bonn. 1998.
[23] Applegate D, Bixby R, Chvátal V, Cook W. The Traveling Salesman Problem: A Computational Study. Princeton, New Jersey, USA: Princeton University Press. 2006.
[24] Fatma A, Karkory A, A Abudalmola. Implementation of Heuristics for Solving Travelling Salesman Problem Using Nearest Neighbour and Minimum Spanning Tree Algorithms. *International Journal of Mathematical, Computational, Natural and Physical Engineering.* 2013; 7(10).
[25] http://en.wikipedia.org/wiki/Travelling_salesman_problem (assessed on 08-06-2015).
[26] Turban E. Decision support and expert system. Macmillan series in information system. 1990.