International Journal of Psychosocial Rehabilitation, Vol. 24, Issue 06, 2020
ISSN: 1475-7192

Optimization of Dynamic Programming Technique for Travelling Salesman Problem

Harshit Dawar, Ninni Singh, Gunjan Chhabra*

Abstract

Travelling Salesman Problem (TSP) is a problem where an optimal path has to be calculated from source city to destination. Optimality of the path generally refers to the minimum cost (distance) between source and destination. In the proposed methodology, multiple optimal paths are calculated which not only provides the user an optimal path, but it also provides the user choice to select an optimal path from the list of multiple optimal paths having same source and destination, but different cities pattern in between them using dynamic programming. The proposed methodology has been compared with the branch and bound approach of solving TSP, and the proposed methodology came out to be faster than the latter approach, and also it provides the multiple optimal paths to the user which is not provided by any other approach.

Keywords: ***T****SP, Dynamic Programming, Optimal Path, Time Complexity, Root City, Initial Root City.*

I. Introduction

The origin of the problem is in 1832 where example tours of Germany and Switzerland are mentioned but proper mathematical explanation and treatment were not present for the problem. TSP belongs to the field of algorithms, where the problem is to find a proper route using a mathematical model which results in an optimal cost for the travel. Before the name TSP is coined, there were too many problems which were present earlier, some of them are, the problem of selecting a route to travel across the country which is optimal/efficient like which is stated at the starting of this introduction column, but there was no proper term or category to which these types of problems can be categorized to. In general, the problem is of movement, that movement can either be of a person, or something else, but that movement should be done in a way that results in an optimal movement path i.e. a path with optimal cost. Then these types of problems have been given a proper name/category that is TSP (Travelling Salesman Problem), by which it became easy to think about these problems and create a solution for them. The very first example of Travelling Salesman Problem was given by Euler in 1759, which is to move a knight exactly once on every position of a chessboard. The very first formulation of this problem was done in 1800 by the Irish mathematician W.R Hamilton, and Thomas Kirkman. The cause of the formulation of this problem was the Hamiltonian's Icosian Game which was based on finding the Hamiltonian cycle.

Mathematically, it was first considered in 1930 by Merrill M. Flood who aims to solve the school bus routing problem.

Travelling Salesman Problem name was first introduced at Princeton University by Hassler Whitney.

TSP is a problem in which a path has to be calculated in which a person should visit each city exactly once given in a scenario and return to the starting city, and the cost of the path should be minimum.

There are many existing techniques to solve TSP, among them the best is dynamic programming which has the Time complexity of $O(n^2 * 2^n)$. This technique can provide the optimal path in the minimum time possible. But the path calculated is not always preferable to users, because the user may want to visit any other in between the routes irrespective of the fact that it may increase the cost of the route. To explain the scenario, consider an example of the path displayed by the algorithm is C_1 -> C_3 -> C_2 -> C_4 -> C_5, where C_1, C_2, C_3, C_4, and C_5 are the cities for the given scenario. The user may want to visit city "B" before "C", but that path is not displayed in this scenario and the user will not be able to know the cost for that path.

Some of the applications of the TSP is as follows:

- **Job Scheduling:** When the jobs are scheduled on a machine, then the time is taken by the machine to complete each job and the time taken by the machine to prepare the machine for the next job is also a TSP.

Department of Computer Science and Engineering, University of Petroleum and Energy Studies, Dehradun, 248001

Received: 18 Jan 2020 | Revised: 09 Feb 2020 | Accepted: 02 Mar 2020

- **Finding an optimal path or route:** To find an optimal path or route which can be required in many situations like transportation is also a TSP.
- **Drilling Problem of Printed Circuit Boards:** While drilling the PCB's, there are many holes which have to be drilled and they are of different diameters, with usual approach, consider two holes of different diameter's have to be drilled consecutively, then the first hole has to be drilled, then change the very time-consuming drilling equipment, but with TSP, all the holes of the same diameter can be drilled in one go, and then change the drilling equipment and then drill the other hole of different diameter, and so on. So, this problem can be referred to as different TSP's combined.
- **Overhauling Gas Turbines in Aircraft:** In an aircraft, its gas turbine engines have to be overhauled, to guarantee the uniform gas flow through the turbines. To fulfill this operation there are nozzle guide vane assemblies located at each turbine stage. The placement of these vanes is considered as a TSP for efficient gas flow through the turbines.
- **Order picking problem in a warehouse:** Various orders of various customers from various platforms arrive in a warehouse and stored at different locations. Each location, if it is considered as a node in a graph, then directly TSP can be implemented there to follow an optimal path for the pickup of orders.
- **Computer wiring:** It is a problem in which a set of pins of different modules has to be connected to the computer board. While connecting is has to be ensured that no more than two pins have to be connected. To do that TSP has to be implemented here.
- **Scheduling Crew problem:** In this problem deposits of different banks has to be picked up from branch banks and returned to the central office by a messenger crew. Routes with minimum cost have to be determined here with overall minimum cost.

This research article addresses the aforementioned issue. A dynamic programming-based solution is proposed that not only provide all possible optimal solutions corresponding to each city except the initial root city from source to destination, in addition to that it will also allow the user to select one path from all the optimal solutions.

II. Background and Preliminaries

This section describes the previous work that provides the optimal solution to TSP. Various research articles have been studied and a few of them are discussed in this section.

In January 2012, [1] D.L Bixby has planned restructuring transport routes according to two methods applied to solve the TSP problems: the nearest neighbor method and the edge classified. The title was "Operations Research Techniques in Management of Samples Transport to the Clinical Laboratory". Results were analyzed that 12% of the paths used the excessive time to reach the laboratory (more than 90 minutes). These routes were modified by applying the nearest neighbor method.

In 2006, [2] Cook W.J, have derived solutions to almost eighty-six thousand cities. The title was "Traveling salesman problem: a computational study". They described the method and computer code they used to solve a broad range of large-scale problems, and along the way, they demonstrate the interplay of applied mathematics with increasingly powerful computing platforms.

In 2011, [3] S. M. Shah, Mahesh Panchal has contributed to the study of various parent selection methods such as Roulette Wheel, Elitism, and Tournament selection methods for Travelling Salesman problem. The title was "Comparison of parents selection methods of genetic algorithm for TSP". And he found that the elitism method is best in all these methods.

In 2014, [4] Rai. K. proposed the Genetic Algorithm which was used to solve the TSP and find an efficient solution. The concept of the Hamiltonian circuit was used to solve it.

In 2010, [5] Matai, presented the study on the topics Applications of TSP, Formulations of TSP and its solutions approaches. Various applications of TSP are explained in it, with its overview formulations and what can be possible approaches for its solution.

In 2013, [6] Sharad N, proposed Firefly algorithm which was used to solve TSP in which there is an objective function, which generates an initial population of fireflies, light intensity at each firefly, then the light absorption coefficient is defined, vary attractiveness with distance r, and then evaluate new solutions and update light intensity.

Received: 18 Jan 2020 | Revised: 09 Feb 2020 | Accepted: 02 Mar 2020

In 1997, [7] Marco proposed the approach of Ant colonies in which there are "m" artificial ants which are placed on some random cities, every time an ant moves, they update the pheromone trail on the edges, and it is known as local trail updating. After all the ants have completed the tour, the ant that made the shortest tour, it is known as global trail updating.

In 2013, [8] Lakshmi G Vijaya, presented the Lexisearch approach to solving TSP, in which every node is considered to be a letter in an alphabet, and each tour is represented as a word with the alphabet. A set of words is known as a block. All the arrangements in a block are separate paths among which the best path is selected.

In 2013, [9] Sudhakara E., presented the Lexisearch approach using a pattern recognition technique. In this scenario, a salesman wants to visit m cities, and m is even. Here N is the set of stations defined, and N1UN2 = N. Salesman has to start from the city which is in N1 and then he visits a city in N2. Here the salesman has to visit N1 and N2 alternatively to get the optimal path.

Kaur, M. [10] presented Multiple travelling salesman problem which has a goal to give the optimal path same as the TSP gives, but in this, there can be more than one salesman involved.

In 1990, [11] Martello S, presented selective TSP, in which a weighted graph is given, which has profits associated with its vertices, the goal of the algorithm is to select a circuit of maximum total profit, whose length does not exceed the specified bound.

III. Proposed Methodology

This section explains the proposed methodology. The proposed methodology is designed to solve the Travelling Salesman Problem using Dynamic Programming, in addition to that, it provides the freedom of choice to the user to choose from the multiple optimal paths.

The multiple optimal paths provided by the proposed methodology to a user, acts as a complete change in the concrete behavior of the existing techniques. It is implemented by applying dynamic programming to each of the individual TSP generated by taking each of the city except Initial Root city from the input matrix as Root city, which leads to multiple paths, among them, the user will have the freedom to choose that path which is suitable to their requirement.

Steps of the proposed methodology are as follows: -

Step 1: Number of cities is requested as input from the user.

Step 2: Distance Matrix of all the cities is requested as input from the user.

Step 3: User is given a choice for selecting the root city from the entered cities.

Step 4: Cost of travelling each city from the root city is calculated and stored in a separate array.

Step 5: From remaining cities (except the root city), select each city as the root city and apply normal dynamic programming solution for TSP on each of them, and store the optimal path obtained from each of city taken as root separately.

Step 6: Cost of traveling from each of the city which is the last city in each of the paths obtained above to the initial root city (for the two-way journey) is calculated and stored separately.

Step 7: The final cost for each path is calculated and stored in a separate array.

Step 8: Display all the paths obtained with their respective cost as the final output.

Note: For visualization of the workflow of the proposed methodology, the aforementioned are illustrated in figure 1.

Figure 1: Flowchart for the Proposed Methodology

IV. Mathematical Model

This section explains the mathematical model for the proposed algorithm.

Table 1. Describes the symbols and their respective definition used for describing the mathematical model.

Symbol	Definition
G	Graph for processing
N	Set of Nodes in the Graph G
E	Set of Edges in the Graph G
i	Starting vertex of any edge
j	Ending vertex of any edge
$C_{i,j}$	Cost of edge (i,j)
R	Initial root city
N - R	Set of edges of graph G excluding the initial root city
x	A Subset of (N - R)

Let us consider a Graph named "G" to explain the mathematical model for the proposed algorithm.

$$G = \{N, E\} \quad (1)$$

Any edge (i, j) $\in$ E will have a cost $C_{i,j}$ associated with it.

Now, let us consider "R" as the initial root city, then according to the proposed algorithm, the remaining graph will now look like,

$$G = \{(N - R), E\} \quad (2)$$

Now, the cost $C_{i,j}$ is calculated for each of the x $\in$ (N - R), and (i, j) $\in$ E from "R" and will be stored in a separate array, let us name that array "first_iteration".

$$first\_iteration = C_{i,j} \ \forall \ path\ starting\ from\ "R"\ to\ x \in (N - R),\ and\ (i, j) \in E \quad (3)$$

Then, for each x ∈ (N - R), the best path starting from x is calculated(bases on the distance) using the dynamic programming approach with their respective costs and stored in a different matrix, let us call it "second_level_matrix".

$$\text{Second\_level\_matrix} = C_{i,j} \quad \forall \text{ best path starting from } x \in (N - R), \text{ and } (i, j) \in E \tag{4}$$

Note: Except Initial Root city, the proposed methodology calculates one optimal path for each other city taking as root city. So, there exists a one-to-many (here many will be (n - 2) cities, where n is the number of cities) mapping in each path, which is presented in Figure 2.

Now, the proposed algorithm will combine the path and their corresponding cost present in the second_level_matrix and first_iteration array, and display each path with their respective cost. This algorithm will provide (n – 1) possible paths considering "n" input cities.

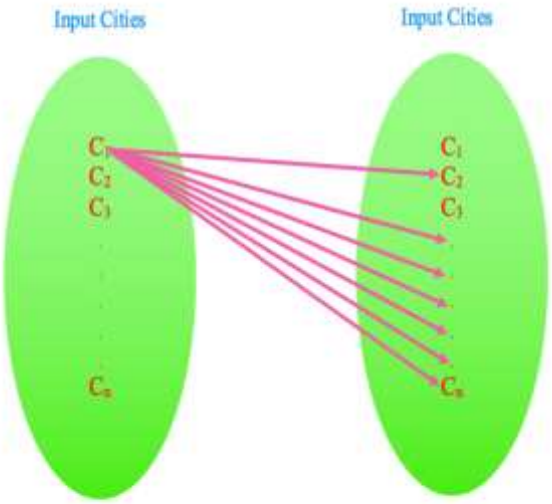

Figure 2: One-to-many Mapping for construction of one path

The figure shown above illustrates the mapping which is performed by the proposed methodology behind the scenes. In the mapping shown above, it is taken into consideration that "C_3" is the initial root city, and mapping for only one path has been illustrated in the figure 2.

Every other path will be constructed using the above mapping only, in the proposed methodology mapping will be done in a way in which each city except the initial root city, will be mapped to (n – 2) other cities to construct the subpath of the complete path, which is then inserted between the block of initial root city. Here, a block of initial root city means that initial root city will be taken as the starting point and ending point of the journey, between this block, the constructed path will be inserted to form a complete path along with its cost, which is then displayed on the output console.

In the mapping, each city is mapped to (n - 2) to other cities because, initial root city is neglected, and also the city which is selected for construction of the path, cannot be mapped to itself in a path. So, two cities are removed from the n cities, which leads to the mapping of one-to-many, where "many" is (n – 2).

From the above mathematical model, it has been concluded that the proposed methodology not only uses dynamic programming for the best Time Complexity of the algorithm, in addition to that it also provides the multiple optimal paths to the user, among which users can choose according to their need.

V. Algorithm and Case Study

This section explains the proposed algorithm and case study for the same.

Algorithm

Input: Distance matrix of the cities
Output: Various best paths starting from each city except the initial root city with their respective costs.

Begin

1. Take the initial root city index in the distance matrix as input from the user, let say "R".

2. Calculate the cost for each other city from initial root city and store it in a separate array, let's say "first_level".
    a. For i from 0 to length(dimension(input_matrix))
        i. {
            1. If (i == R){
                a. Pass
            2. }
            3. Else {
                a. Firstlevel[i] <- distance_matrix[R][i]
            4. }
        ii. }

3. A "Level2Matrix" is created which will be the same as the distance matrix but the only difference will be that the initial root matrix will not be present in this matrix.

4. Now for each city except the initial root city will be taken as root city and the dynamic programming approach is used to solve TSP starting from that city, it is done by declaring an array initially all with zeros with a size equal to the dimension of the distance matrix, which will be used to check that whether a particular city has been visited or not, and if not, then the particular index in that array will be changed to 1. Also, the next city is selected on the bases of the minimum cost.
    a. For i in remaining_cities
        i. {
            1. Mincost <- some minimum infinite value
            2. Nextcity <- random number
            3. Starting_city <- assigned from the level2matrix
            4. If (level2matrix[i][city] != 0 and visited[i] != 0)
                a. {
                    i. If (level2matrix[city][i] < mincost)
                        1. {
                            a. Nextcity = i
                        2. }
                b. }
        ii. }
5. The above logic will be called in a separate function which will be then used to construct the best path from each city as root city present in the level2matrix.

6. All the calculated paths will be displayed on the output console out of which the user has a choice of a path optimal according to it.

End

Received: 18 Jan 2020 | Revised: 09 Feb 2020 | Accepted: 02 Mar 2020

Table 2: Case Study for the Algorithm

Input	Intermediate	Output
Suppose the distance matrix containing 3 cities is taken as input [Delhi, Chandigarh, and Kashmir] as [0, 265, 583] [265, 0, 931] [583, 931, 0] respectively, and given Delhi as the initial root city.	The proposed algorithm will take "Delhi" as the initial root and start finding best the path from "Kashmir" and "Chandigarh" which will route back to Delhi, and their respective costs.	All the possible output paths are presented in Figure 3.

Figure 3: Output of the Case Study

VI. Results and Analysis

This section illustrates the comparative study between the proposed methodology and the existing technique branch & bound. Both of them are implemented to solve the Travelling Salesman Problem, and their results have been observed.

To perform a comparative analysis of both of the techniques i.e. the proposed methodology and the branch & bound technique, worst-case Time Complexity and the number of output paths for each of the techniques are taken as a factor for comparison.

Branch & bound technique has the Time Complexity equivalent to the Time Complexity of the brute force approach, and it provides only one output path to the user. On the other hand, the proposed methodology has the Time Complexity of $O(n^2 * 2^n)$, which is very less as compared to the branch & bound technique. Also, the proposed methodology provides (n - 1) output paths to the user to choose from, where n is the number of input cities.

In the implementation of both of the techniques, the same dataset has been used. Result of both the techniques are listed in Table 3, and graphically presented in Figure 4. Execution time which is taken as an attribute in Table 3 represents the time taken by the algorithm to run which is also known as Time Complexity in general terms.

Table 3: Execution Time for both the algorithms

No. of Cities	Time of Execution(in seconds) for the Proposed Algorithm	Time of Execution(in seconds) for Branch & Bound Algorithm
3	0.000593	0.000995
4	0.001027	0.001359
5	0.001078	0.001975
6	0.001123	0.002546
7	0.001845	0.003017

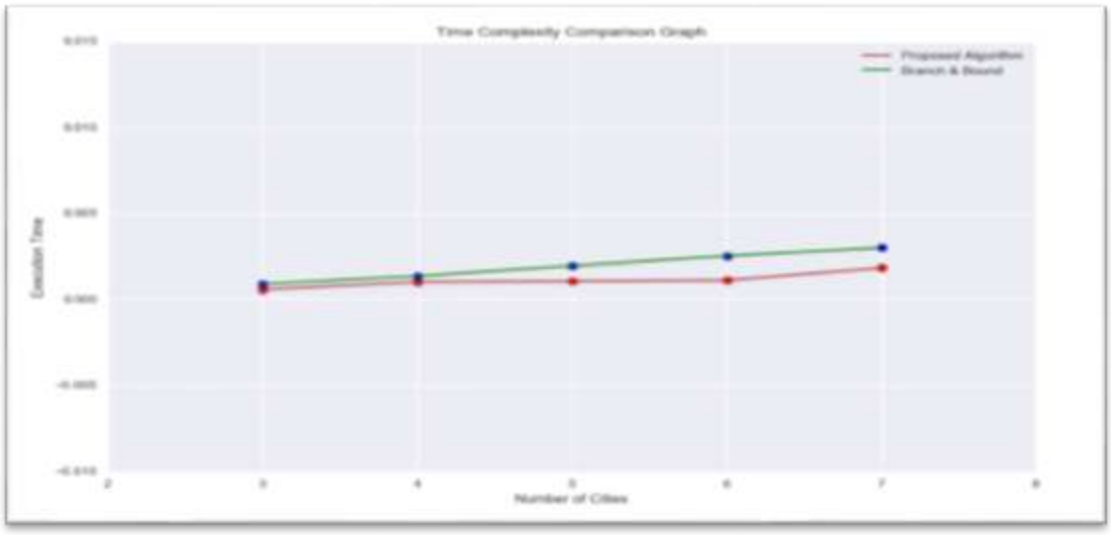

Figure 4: Graph for Execution Time comparison for both the algorithms

By analyzing the Figure 4, and Table 3 which contains the execution time for both of the techniques, it is observed that the proposed methodology took less execution time, and in addition to that it is clear from the beginning of this research paper that, the proposed methodology provides multiple output paths to the user.
Therefore, on both the basis used for the comparison of the proposed methodology and the branch & bound technique, the proposed methodology performs better than the branch & bound, and proves to be a better technique to handle TSP in less time, as well as provide more choices to the users.
Benefits of the proposed algorithm are as follows: -

a) It provides flexibility to the user for selecting the path which the user wants.
b) No compulsion on the user for selecting the proposed optimal path only.
c) Provides a wider view of all the possibilities in the given scenario.
d) It Takes less running time than other existing techniques although it has to calculate multiple paths.

VII. Conclusion

Proposed methodology proves to not only take the least execution time as compared to other technique branch & bound, but also it provides multiple paths among which users of the algorithm have a choice. With the input of 3 cities, the branch & bound technique took 0.000995 seconds, whereas the proposed methodology took just 0.000593 seconds to complete. In the case of 4 cities, the branch & bound technique took 0.001359 seconds, whereas the proposed methodology took just 0.001027 seconds to complete. In the case of 5 cities, the branch & bound technique took 0.001975 seconds, whereas the proposed methodology took just 0.001078 seconds to complete. In the case of 6 cities, the branch & bound technique took 0.002546 seconds, whereas the proposed methodology took just 0.001123 seconds to complete. In the case of 7 cities, the branch & bound technique took 0.003017 seconds, whereas the proposed methodology took just 0.001845 seconds to complete. From the results, it has been concluded that the proposed methodology is a better technique/approach to solve the Travelling Salesman Problem which benefits its users in every aspect.

References

[1] Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J. (2007). The Traveling Salesman Problem. Princeton University Press.
[2] Applegate DL, Bixby RE, Chvatal V, Cook WJ. The traveling salesman problem: a computational study. Princeton university press; 2006.

Received: 18 Jan 2020 | Revised: 09 Feb 2020 | Accepted: 02 Mar 2020

[3] Chudasama, C., Shah, S. M., & Panchal, M. (2011). Comparison of parents selection methods of genetic algorithm for TSP. In *International Conference on Computer Communication and Networks CSI-COMNET-2011, Proceedings* (pp. 85-87).
[4] Rai, K., Madan, L., & Anand, K. (2014). Research Paper on Travelling Salesman Problem And it's Solution Using Genetic Algorithm. *International Journal of Innovative Research In Technology*, *1*(11), 103-114.
[5] Matai, Rajesh, Surya Prakash Singh, and Murari Lal Mittal. "Traveling salesman problem: an overview of applications, formulations, and solution approaches." *Traveling salesman problem, theory, and applications* 1 (2010).
[6] Kumbharana, Sharad N., and Gopal M. Pandey. "Solving travelling salesman problem using firefly algorithm." *International Journal for Research in Science & advanced Technologies* 2.2 (2013): 53-57.
[7] Dorigo, Marco, and Luca Maria Gambardella. "Ant colonies for the travelling salesman problem." *biosystems* 43.2 (1997): 73-81.
[8] Lakshmi, G. Vijaya. "Lexisearch Approach to Travelling Salesman Problem."
[9] Reddy, P. M. M., Sudhakara, E., Sreenadh, S., & Varma, S. V. K. An Alternate Travelling Salesman Problem.
[10] Singh, A., & Kaur, M. Solving Multiple Travelling Salesman Problem Using Genetic Algorithm: A Case Study.
[11] Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. *Discrete applied mathematics*, *26*(2-3), 193-207.