Asian Journal of Probability and Statistics

Volume 26, Issue 6, Page 63-72, 2024; Article no.AJPAS.118144
ISSN: 2582-0230

Travel Salesman Problem Using Dynamic Programming

Anosike Joseph Ugonna [a], Amagoh Maureen Nkechi [b] and Orumie Ukamaka Cynthia [a*]

[a] University of Port Harcourt Nigeria, Nigeria.
[b] Delta State Polytechnic Ogwashi -Uku, Delta State, Nigeria.

Authors' contributions

This work was carried out in collaboration among all authors. All authors read and approved the final manuscript.

Article Information

DOI: https://doi.org/10.9734/ajpas/2024/v26i6625

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://www.sdiarticle5.com/review-history/118144

Original Research Article

Received: 22/03/2024
Accepted: 31/05/2024
Published: 04/06/2024

Abstract

In the fields of operations research and computer science, the traveling salesman problem (also known as TSP) is a classic algorithmic issue. One method for solving TSP is dynamic programming, which identifies the optimal path at the lowest possible cost. It starts by determining the shortest path between two points, and then it expands to identify routes to other sites. In order to solve the Travel salesman Problem, Dynamic Programming was used in this study as a a solution technique. In addition to a real-world application from AM-EXPRESS NIGERIA LIMITED, illustrative examples of travel salesman difficulties were taken into consideration and handled utilizing the dynamic programming technique. The finding's outcome showed that the least amount of money needed to provide logistical services is ₦6500, and the shortest possible paths or routes were found.

**Corresponding author: Email: Ukamaka.orumie@uniport.edu.ng;*

Cite as: *Ugonna, Anosike Joseph, Amagoh Maureen Nkechi, and Orumie Ukamaka Cynthia. 2024. "Travel Salesman Problem Using Dynamic Programming". Asian Journal of Probability and Statistics 26 (6):63-72. https://doi.org/10.9734/ajpas/2024/v26i6625.*

Keywords: Travel salesman problem; dynamic programming; minimum path; route.

1 Introduction

In computer science and operations research, the traveling salesman problem, or TSP for short, is a traditional algorithmic problem. In this case, a quicker, shorter, or less expensive alternative is frequently considered better. The salesman in this problem has to go between N cities. As long as he stays at his starting location and doesn't visit a city more than once while traveling, the order in which he goes is irrelevant. Finding the quickest or least expensive path for a salesperson to travel in order to visit a number of places in one trip and then return to the beginning point is the main goal of the challenge [1-3].

There is pressure on a salesperson to locate the best and shortest route when they are out in the field visiting different client sites. Nowadays, there are other factors besides distance that determine efficiency. Efficiency is determined by a number of elements taken together, including capacity, fuel usage, time, and others. But resolving the issues with traveling salesmen also means lessening some of the inevitable obstacles that field agents must overcome [4-6].

One method for resolving TSP is dynamic programming, which is similar to taking a jigsaw and gradually putting it together piece by piece. It determines the most efficient path to visit every place in TSP tasks. Finding the shortest path between two locations is the first step, and it then expands to identify routes to more locations. It's a clever TSP solution for little situations, but for bigger and more complicated issues, it could need a lot of RAM [7-9].

Numerous studies on travel salesmen have been conducted. A dynamic programming technique was used by Arifin and Yusuf [10] to develop a replacement model for city bus vehicles used in Bandung. Cases that Damri Company experienced while attempting to enhance public services served as the impetus for their investigation. Their replacement model offered two options for policy: either keep the current vehicles or replace them with new ones while accounting for new car purchase expenses, salvage value, and operational costs. The findings indicated that a bus's economic life is only nine years old, while its technical life is roughly twenty years.

The heterogenous selection evolutionary algorithm (HeSEA), proposed by Long et al. in 2004, is an evolutionary method designed to solve huge TSPs. They initially examined the advantages and disadvantages of several popular genetic operators, as well as local search techniques for TSPs based on the characteristics of their solutions and edge-preserving and edge-adding mechanisms. Based on that research, they developed the HeSEA method, which combines heterogeneous pairing selection and family competition to incorporate Edge Assembly Crossover (EAX) and Lin-Kernighan (LK) local search.

For the solution of TSP, Rajan and Anilkumar [11] suggested a novel search algorithm inspired by the evolutionary optimization technique. With the availability of quick computer resources, they presented novel crossover and mutation operators that are appropriate for using genetic algorithms to solve TSPs. This search method can be applied to solve TSPs with a lot of dimensions.

An algorithm combining a candidate list approach with dynamic heuristic parameter upgrading for local search solution was proposed by Zar and May [12]. A static list contains a number of favored nodes that are kept in the dynamic candidate list. An ant chooses the node that is listed in the favored list as it goes from one node to the next. This approach is used to implement an ant colony search algorithm on larger datasets. The upgrading was predicated on the emergence of solutions and entropy. Their proposed approach is significantly more effective in terms of speed and capacity to find superior answers, based on the outcomes of their experiments.

In order to solve the shortest path problem, Alwan [13] suggested a systolic parallel system that used simultaneous forward and backward dynamic programming. In applications, the systolic solution's speed advantage was crucial, particularly for shortest path routing in wireless networks. The approach was valued in cases where the directed acyclic graph (DAG) with an odd number of stages—though the case with an even number of stages was also taken into consideration.

In 2013, Saloni and Poonam introduced the idea of using genetic algorithms to solve the traveling salesman problem. The effectiveness of these algorithms relies heavily on the encoding of the problem as well as the crossover and mutation techniques applied. Several genetic algorithm techniques have been examined and evaluated in order to solve TSP. Further investigation into alternative hybrid selection, crossover, and mutation operators is possible. Numerous advanced network models, including logistic networks, job scheduling models, and vehicle navigation routing models, can be used with the suggested methodology. The distribution of frequencies among cellular network cells can likewise be accomplished using the same methodology.

In their research, Sonam and Puneet (2014) came to the conclusion that the traveling salesman problem can be resolved using genetic algorithms. Depending on the encoding of the issue and the kinds of crossover and mutation techniques applied, a genetic algorithm can determine an optimal solution for the TSP. Many genetic algorithm methods for resolving TSP have been examined and analyzed.

Wu [14] investigated how production planning uses dynamic programming. According to him, dynamic programming is a technique that breaks down a big problem into a number of smaller, more manageable subproblems, solves each of the smaller problems just once, and then stores the solutions. According to him, the optimal sub-structure, the non-after effect property, and the overlapping subproblems are the three key characteristics of the problems that determine how effective the dynamic programming technique is. He came to the conclusion that, when compared to other methodologies, the dynamic programming technique is particularly effective in reducing computational complexity and improving computational outcomes. He created a production planning problem model using dynamic programming.

As a result, the researchers want to use dynamic programming to solve the travel salesman problem with an exemplary case. Their goals are to: Create a recursive tree, Utilizing the traveling salesman problem's generic formula, Determine the best price, and then determine the best routes.

In their study, Souhail et al. [15] presented a novel greedy technique, named Dhouib-Matrix-TSP1, as the first resolution of TSP to obtain the optimal solution with multiple numerical examples utilizing single valued trapezoidal neutrosophic numbers. Graphical solutions were also used for analysis of the outcomes.

In their article, Souhail [16] introduced a novel metaheuristic called Far-to-Near (FtN) for the purpose of maximizing the shortest path for drilling holes in printed circuit boards (PCBs). The produced solution served as the robotic arm's CNCM execution plan. The effectiveness of the suggested metaheuristic FtN in generating the shortest drilling route and determining the best circuit when adjusting the robotic arm's initial starting point was demonstrated by a number of numerical examples.

In their research, Souhail and Zouari [17] proposed a novel method for drilling a hole series while optimizing non-productive tool paths. The iterated stochastic Dhouib-Matrix-3 (DM3) was combined with a tabu memory that was influenced by the Tabu Search (TS) metaheuristic to form their suggested method, which they called Adaptive-Dhouib-Matrix-3 (A-DM3). Six real-world case studies involving a rectangular matrix of holes were used to test their methodology. Their approach was compared to a few widely used algorithms, including the hybrid Cuckoo Search Genetic Algorithm (CS-GA), modified Shuffled Frog Leaping Algorithm (mSFLA), Ant Colony Optimization (ACO) and some of its derivatives, Genetic Algorithm (GA), and Cuckoo Search (CS). Their suggested A-DM3 outperformed these well-known metaheuristics in the literature, especially in a medium and large number of holes, as demonstrated by computational data. As a result, A-DM3 beat rival algorithms to produce a new record for the shortest path length, often improving upon it by almost 100%.

The goal of Souhail et al.'s [18] study was to identify a collection of effective solutions for the multi-objective Traveling Salesman Problem. A novel metaheuristic called DM4-PMO was proposed to obtain the solutions of the Pareto set. The two steps of the DM4-PMO are as follows: First, the first non-dominated pareto frontier solutions were found using a weighted sum function with many variations of weights. To create the final Pareto frontier solutions, a lexicographical resolution was used on a few non-dominated solutions from the initial Pareto frontier. An experiment using two-objective problems from the TSP-LIB and DIMACS datasets proved the effectiveness of the suggested method [19,20].

The test results demonstrated the robustness, speed, and simplicity of structure of the suggested DM4-PMO, which can quickly compute a Pareto non-dominated set solution with a minimal number of user-defined parameters [21].

2 Scope of the Study

The Traveling Salesman Problem (TSP) can be solved with a thorough understanding of the dynamic programming technique provided by this paper. It offers AMEXPRRESS NIGERIA LIMITED, a small logistics company with its headquarters in Lagos, Nigeria, a solution to its transportation cost problems.

The real-world example's data set came from five (5) different delivery destinations that were managed by AMEXPRESS NIG. LTD. Ikeja serves as the source city while the other four cities are Apapa, Surulere, Oshodi, Ikorodu, and Ikeja.

3 Methodology

3.1 Terminology in DPP

Terminology commonly used in dynamic programming are given below:

Stage: A stage is the moment at which a choice is made. A stage ends and the next one that follows immediately begins. A stage ends and the next one that follows immediately begins. For example, in the resource allocation to shops problem, each shop represents a stage; in the salesman allocation problem, each territory represents a stage.

State: A state variable is the variable that connects two stages of a multistage decision issue. The problem's current state is described by the values that state variables can have at any given time. We refer to these values as states. For example, resource is a state variable when it comes to resource allocation to shops.

Stage decision: Every stage has several options; the process of choosing one of the best and most workable options is known as the stage decision.

Principle of optimality: The principle of optimality state that that the optimal decision from any state in a stage to the end, is independent of how one actually arrives at that state.

Optimal policy: A policy which optimizes the value of an objective function.

Return function: Every step involves making a decision that can impact the system's condition in the following step and aid in finding the best solution for the situation at hand. Each choice has a benefit that may be expressed as an algebraic equation. We refer to the equation as a return function.

3.2 Dynamic programming

Using this method, one can resolve a multi-stage decision problem where choices must be made at each level. It lacks a universally applicable formulation. In other words, a recursive equation that fits the circumstances must be created. Richard Bellman invented this method in 1957. Bellman's principle is the basis for this model's development.

The following is an explanation of this principle: An optimal policy has the property that all subsequent decisions must be optimal in relation to the state that results from the initial decision, regardless of the original state and decisions made.

The foundation of the dynamic programming technique is the aforementioned principle. When creating a recursive equation, the Bellman's principle must be taken into account. Recursive equations are founded on the idea that a strategy is "optimal" if decisions taken at each step lead to overall optimality over all stages, not just the current one. They express subsequent state conditions in terms of the processed state circumstances.

By saving and reusing the solutions to subproblems, dynamic programming aims to prevent repeatedly computing the same problems. In this approach, dynamic programming can frequently identify the best solution to a problem while lowering the time and space complexity of recursive methods. The traveling salesman problem's weighted graph G is shown below;

$$G = (V, E),$$

Where v implies a vertex set (cities) and E implies set of edges fully connected with the nodes. Each edge $(i, j) \in E$ is assigned to a weight $d_{i,j}$ which represents the distance between i and j.

$$cost,\ C(i,j) = \begin{cases} 0 \ \ if\ i = j \\ cost\ \ if\ (i,j) \in \mathrm{E} \\ \infty \ \ if\ (i,j) \notin E \end{cases}$$

3.2.1 Constraints for TSP

Time constraints: sales agent often have a tight schedule with multiple deliveries to make with a short TAT. Similarly, in TSP, optimizing routes to minimize travel time is a fundamental challenge.

Resource efficiency: Just as salesperson aim at reducing costs and ensuring on-time deliveries, TSP solutions must stive for resource optimization by reducing travel distances and delivery TAT.

Objective Diversification: While solving the travelling salesman problem (TSP), optimizing multiple objectives such as cost, time and environmental factors adds complexity as solutions need balance conflicting goals.

3.3 Rules of thumb with dynamic programming

Subproblems overlap: Subproblems are smaller variations of an original, larger problem. for example, in the Fibonacci sequence, each number in the series is the sum of its two preceding numbers (0,1,1,2,3,5,8, 13…). If you want to calculate the nth Fibonacci value in the sequence, you can break down the entire problems into subproblems that overlap with one another as you find solutions by solving same subproblems repeatedly.

Substructure has optimal property: When you can construct an optimal solution after constructing all other solutions that resulted from every subproblem you addressed, the optimal substructure property materializes. Each overlap's solution is applicable to the entire issue in order to maximize recursion. Each subproblem in the Fibonacci sequence example has a solution that can be applied to the next subproblem to determine the number in the series, giving the problem as a whole the best substructure property.

3.4 General characteristics of dynamic programming

- The problem is divided into stages
- Each stage has a number of states associated with it.
- Making decisions at one stage transforms one state of the current stage into a state in the next stage.
- Given the current state, the optimal decision for each of the remaining states does not depend on the previous states or decisions. This is known as the principle of optimality for dynamic programming.
- The principle of optimality allows to solve the problems stage by stage recursively.
- Construct a recursive equation to make a decision based on the optimum policy at each stage.
- After the construction of the recursive equation, regardless the solution, either we can move forward or backward.
- Dynamic programming allows for both forward and backward recursion for computation, with the best answer to one subproblem being utilized as an input for the next. By the time the final subproblem is resolved, the ideal solution for the whole issue is nearly attained.

3.5 Dynamic programming algorithm

1. Identify the decision variables and specify objective function to be optimized under certain limitations if any

2. Divide the given sub problem into a number of stages
3. Identify the state variables at each stage and write down the transformation function as a function of state variable and decision variable at the next stage.
4. Decide whether forward or backward method is to be followed to solve the problem
5. Determine the overall optimal policy or decisions and its value at each stage.

The dynamic programming cost function in relation to TPP is given below;

$$\boldsymbol{g}(\boldsymbol{i},\boldsymbol{j}) = \min_{\boldsymbol{j}\in\boldsymbol{S}}\{\boldsymbol{C}_{\boldsymbol{ij}} + \boldsymbol{g}(\boldsymbol{j},\boldsymbol{s} - \{\boldsymbol{j}\})$$

Where i is the starting vertex, j is the current vertex travelled to, and S is the set of remaining vertices.

3.6 Real-life application

The following table represents the Pick-up points and travel cost (Naira) incurred by AM-EXPRESS NIGERIA LIMITED in its operations.

We desire the determine the paths of least cost and the least cost in carrying out her operations.

Table 1. The pick-up points and travel cost (Naira) incurred by AM-EXPRESS Nigeria Limited

	Ikeja	**Surulere**	**Oshodi**	**Apapa**	**Ikorodu**
Ikeja	0	1000	700	1200	1800
Surulere	1000	0	800	1500	2500
Oshodi	700	800	0	1000	1200
Apapa	1200	1500	1000	0	3000
Ikorodu	1800	2500	1200	3000	0

3.7 Analyses

Table 2: The cost adjacency matrix from Table1.
The cost adjacency matrix is given below;

$$A = \begin{array}{c|ccccc} & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \mathbf{5} \\ \hline \mathbf{1} & \mathbf{0} & \mathbf{1000} & \mathbf{700} & \mathbf{1200} & \mathbf{1800} \\ \mathbf{2} & \mathbf{1000} & \mathbf{0} & \mathbf{800} & \mathbf{1500} & \mathbf{2500} \\ \mathbf{3} & \mathbf{700} & \mathbf{800} & \mathbf{0} & \mathbf{1000} & \mathbf{1200} \\ \mathbf{4} & \mathbf{1200} & \mathbf{1500} & \mathbf{1000} & \mathbf{0} & \mathbf{3000} \\ \mathbf{5} & \mathbf{1800} & \mathbf{2500} & \mathbf{1200} & \mathbf{3000} & \mathbf{0} \end{array}$$

For |S| = 0

$g(x,y)$

$g(1,\emptyset) = 0$

$g(2,\emptyset) = 1000$

$g(3,\emptyset) = 700$

$g(4,\emptyset) = 1200$

$g(5,\emptyset) = 1800$

For |S| = 1

$g(2,\{3\}) = c_{23} + g(3,\emptyset) = 800 + 700 = 1500$

$g(2,\{4\}) = c_{24} + g(4,\emptyset) = 1500 + 1200 = 2700$

$g(2,\{5\}) = c_{25} + g(5,\emptyset) = 2500 + 1800 = 4300$

$g(3,\{2\}) = c_{32} + g(2,\emptyset) = 800 + 1000 = 1800$

$g(3,\{4\}) = c_{34} + g(4,\emptyset) = 1000 + 1200 = 2200$
$g(3,\{5\}) = c_{35} + g(5,\emptyset) = 1200 + 1800 = 3000$
$g(4,\{2\}) = c_{42} + g(2,\emptyset) = 1500 + 1000 = 2500$
$g(4,\{3\}) = c_{43} + g(3,\emptyset) = 1000 + 700 = 1700$
$g(4,\{5\}) = c_{45} + g(5,\emptyset) = 3000 + 1800 = 4800$
$g(5,\{2\}) = c_{52} + g(2,\emptyset) = 2500 + 1000 = 3500$
$g(5,\{3\}) = c_{53} + g(3,\emptyset) = 1200 + 700 = 1900$
$g(5,\{4\}) = c_{54} + g(4,\emptyset) = 3000 + 1200 = 4200$

For |S| = 2

$$g(2,\{3,4\}) = min\begin{cases} c_{23} + g(3,\{4\}) = 800 + 2200 = 3000 \\ c_{24} + g(4,\{3\}) = 1500 + 1700 = 3200 \end{cases} = 3000$$

$$g(2,\{3,5\}) = min\begin{cases} c_{23} + g(3,\{5\}) = 800 + 3000 = 3800 \\ c_{25} + g(5,\{3\}) = 2500 + 1900 = 4400 \end{cases} = 3800$$

$$g(2,\{4,5\}) = min\begin{cases} c_{24} + g(4,\{5\}) = 1500 + 4800 = 6300 \\ c_{25} + g(5,\{4\}) = 2500 + 4200 = 6700 \end{cases} = 6300$$

$$g(3,\{2,4\}) = min\begin{cases} c_{32} + g(2,\{4\}) = 800 + 2700 = 3500 \\ c_{34} + g(4,\{2\}) = 1000 + 2500 = 3500 \end{cases} = 3500$$

$$g(3,\{2,5\}) = min\begin{cases} c_{32} + g(2,\{5\}) = 800 + 4300 = 5100 \\ c_{35} + g(5,\{2\}) = 1200 + 3500 = 4700 \end{cases} = 4700$$

$$g(3,\{4,5\}) = min\begin{cases} c_{34} + g(4,\{5\}) = 1000 + 4800 = 5800 \\ c_{35} + g(5,\{4\}) = 1200 + 4200 = 5400 \end{cases} = 5400$$

$$g(4,\{2,3\}) = min\begin{cases} c_{42} + g(2,\{3\}) = 1500 + 1500 = 3000 \\ c_{43} + g(3,\{2\}) = 1000 + 1800 = 2800 \end{cases} = 2800$$

$$g(4,\{2,5\}) = min\begin{cases} c_{42} + g(2,\{5\}) = 1500 + 4300 = 5800 \\ c_{45} + g(5,\{2\}) = 3000 + 3500 = 6500 \end{cases} = 5800$$

$$g(4,\{3,5\}) = min\begin{cases} c_{43} + g(3,\{5\}) = 1000 + 3000 = 4000 \\ c_{45} + g(5,\{3\}) = 3000 + 1900 = 4900 \end{cases} = 4000$$

$$g(5,\{2,3\}) = min\begin{cases} c_{52} + g(2,\{3\}) = 2500 + 1500 = 4000 \\ c_{53} + g(3,\{2\}) = 1200 + 1800 = 3000 \end{cases} = 3000$$

$$g(5,\{2,4\}) = min\begin{cases} c_{52} + g(2,\{4\}) = 2500 + 2700 = 5200 \\ c_{54} + g(4,\{2\}) = 3000 + 2500 = 5500 \end{cases} = 5200$$

$$g(5,\{3,4\}) = min\begin{cases} c_{53} + g(3,\{4\}) = 1200 + 2200 = 3400 \\ c_{54} + g(4,\{3\}) = 3000 + 1700 = 4700 \end{cases} = 3400$$

For |S| = 3

$$g(2,\{3,4,5\}) = min\begin{cases} c_{23} + g(3,\{4,5\}) = 800 + 5400 = 6200 \\ c_{24} + g(4,\{3,5\}) = 1500 + 4000 = 5500 \\ c_{25} + g(5,\{3,4\}\}) = 2500 + 3400 = 5900 \end{cases} = 5500$$

$$g(3,\{2,4,5\}) = min\begin{cases} c_{32} + g(2,\{4,5\}) = 800 + 6300 = 7100 \\ c_{34} + g(4,\{2,5\}) = 1000 + 5800 = 6800 \\ c_{35} + g(5,\{2,4\}\}) = 1200 + 5200 = 6400 \end{cases} = 6400$$

$$g(4,\{2,3,5\}) = min\begin{cases} c_{42} + g(2,\{3,5\}) = 1500 + 3800 = 5300 \\ c_{43} + g(3,\{2,5\}) = 1000 + 4700 = 5700 \\ c_{45} + g(5,\{2,3\}\}) = 3000 + 3000 = 6000 \end{cases} = 5300$$

$$g(5,\{2,3,4\}) = min\begin{cases} c_{52} + g(2,\{3,4\}) = 2500 + 3000 = 5500 \\ c_{53} + g(3,\{2,4\}) = 1200 + 3500 = 4700 \\ c_{54} + g(4,\{2,3\}\}) = 3000 + 2800 = 5800 \end{cases} = 5500$$

For |S| = 4

$$g(1,\{2,3,4,5\}) = min\begin{cases} c_{12} + g(2,\{3,4,5\}) = 1000 + 5500 = 6500 \\ c_{13} + g(3,\{2,4,5\}) = 700 + 6400 = 7100 \\ c_{14} + g(4,\{2,3,5\}) = 1200 + 5300 = 6500 \\ c_{15} + g(5,\{2,3,4\}) = 1800 + 4700 = 6500 \end{cases} = 6500$$

Minimum cost = ₦6500

Paths: 1→2→4→3→5→1 (Ikeja to Surulere to Apapa to Oshodi to Ikorodu)
OR
1→4→2→3→5→1(Ikeja to Apapa to Surulere to Oshodi to Ikorodu)
OR
1→5→3→2→4→1(Ikeja to Ikorodu to Oshodi to Surulere to Apapa)
OR
1→5→3→4→2→1(Ikeja to Ikorodu to Oshodi to Apapa to Surulere)

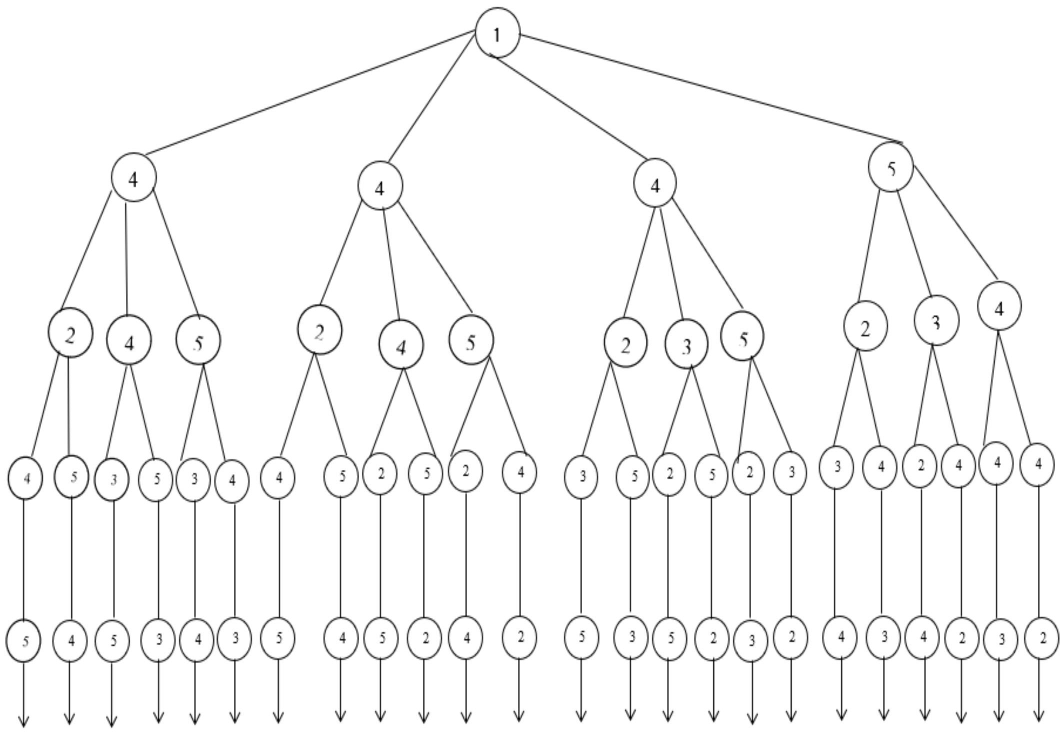

Fig. 1. Recursive tree

4 Summary and Conclusion

In this paper, the travel salesman problem was investigated and solved through the use of dynamic programming. After the real-world application was resolved, it was discovered that the lowest cost of providing logistical services is ₦6500, and the shortest path or route was determined to be as follows:

- Ikeja (1) to Surulere (2) to Apapa (4) to Oshodi (3) to Ikorodu (5) and back to Ikeja
- Ikeja (1) to Apapa (4) to Surulere (2) to Oshodi (3) to Ikorodu (5) and back to Ikeja
- Ikeja (1) to Ikorodu (5) to Oshodi (3) to Surulere (2) to Apapa (4)
- Ikeja (1) to Ikorodu (5) to Oshodi (3) to Apapa (4) to Surulere (3)

Competing Interests

Authors have declared that no competing interests exist.

References

[1] Basu S, Ghosh D. A Review of the Tabu search literature on Traveling Salesman problems. Indian institute of management. 2008;10(1):1-16

[2] Basu S. Tabu search implementation on Traveling Salesman Problem and its variations: A literature survey. American Journal of operations research. 2012;2:163-173.

[3] Bland RE, DE Shallcross Large Traveling Salesman Problem Arising from Experiments in X-Ray Crystallography: A Preliminary Report On Computation. Operations Research Letters. 1989;8(3):125-128.

[4] Hingrajiya KH, Gupta RK, Chandel GS. An Ant Colony Optimization Algorithm. International Journal of Scientific and Research Publications. 2012;2(8):1-6.

[5] Jin L, Tsai S, Yang J, Kao C. An evolutionary algorithm for large travelling Salesman problems. IEEE Transactions on systems, Man and Cybernetics-Part B: Cybernetics. 2004;34(4):1718-29.

[6] Khatter S, Gosawmi P. A Solution of Genetic Algorithm for Solving Traveling Salesman Problem. International Journal for Scientific Research and Development (IJSRD). 2014;2(4):341-3.

[7] Kumar N, Karambir, Kumar R. A Genetic Algorithm Approach to Study Traveling Salesman Problem. Journal Of Global Research in Computer Science. 2012;3(3):33-8.

[8] Plante RD, TJ. Lowe and R. Chandrasekaran the Product Matrix Traveling Salesman Problem: An Application and solution Heuristics. Operation Research. 1987;35:772-783.

[9] Sancho NGF. A Dynamic Programming Solution of a Shortest Path Problem with Time Constraints on Movement and Parking. Journal Of Mathematical Analysis and Applications. 1992;166:192-198.

[10] Arifin D, Yusuf E. (2017). Replacement Model of City Bus; A Dynamic Programming Approach. Conference Proceedings of American Institute of Physics. 2017;020006-1-020006-6.
DOI 10.1063/1.4985451

[11] Rajan K, Anilkumar. Genetically motivated search algorithm for solving Traveling Salesman Problem. SB Academic Review. 2009;XVI(1&2):19- 31.

[12] Zar Chi Su SuHlaing and May Aye Khine, Solving Traveling Salesman by Using Improved Ant Colony Optimization Algorithm, International Journal of Information and Education Technology. 2011;1:5.

[13] Alwan N. Fast computation of the shortest path problem through Simultaneous forward and Backward Sytolic Dynamic programming. International Journal of computer Application. 2012;54(1):21-25.

[14] Run Wu. The application of Dynamic Programming in Production Planning Conference Proceedings of American Institute of Physics. 2017;020155-1-020155-3.
DOI 10.1063/1.4982520.

[15] Souhail Dhouib, Said Broumi M. Lathamaheswari Single Valued Trapezoidal Neutrosophic Travelling Salesman Problem with Novel Greedy Method: The Dhouib-Matrix-TSP1 (DM-TSP1. International Journal of Neutrosophic Science (IJNS). 2021;17(2):144-157.

[16] Souhail Dhouib Hole Drilling Route Optimization in Printed Circuit Boards Using Far-to-Near Metaheuristics: Optimizing the Hole Drilling Route via Far-to-Near Metaheuristic International Journal of Strategic Engineering (IJoSE). 2022;5(1):12.

[17] Souhail Dhouib, Alaeddine Zouari. Adaptive iterated stochastic metaheuristic to optimize holes drilling path in manufacturing industry: The Adaptive-Dhouib-Matrix-3 (A-DM3), Engineering Applications of Artificial Intelligence. 2023;120.

[18] Souhail Dhouib, Aïda Kharrat, Taicir Loukil, Habib Chabchoub Enhancing the Dhouib-Matrix-4 metaheuristic to generate the Pareto non-dominated set solutions for multi-objective travelling salesman problem: The DM4-PMO method, Results in Control and Optimization. 2024;14:100402.
ISSN 2666-7207

[19] Amiens EO, Oisamoje MD, Inegbenebor AU. A Dynamic Programming Approach to Replacement of Transport Vehicles in Benin City, Nigeria. J. Adv. Math. Com. Sci. [Internet]. 2015 Jan. 3 [cited 2024 May 25];6(3):204-1.
Available:https://journaljamcs.com/index.php/JAMCS/article/view/886

[20] Habibi R. Stochastic Dominance, Dynamic Programming and Bayes Filtering: Applications in NPV and NPVaR. Asian J. Econ. Busin. Acc. [Internet]. 2017 Jan. 6 [cited 2024 May 25;2(1):1-7.
Available:https://journalajeba.com/index.php/AJEBA/article/view/152

[21] Salii Y. Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization. European Journal of Operational Research. 2019;272(1):32-42.