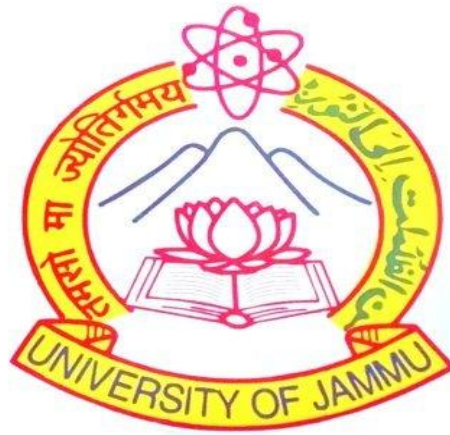


Travelling Salesperson Problem



MAJOR PROJECT REPORT

Semester - 1

Four Year Undergraduate Program-Design your Degree

Submitted to

University of Jammu, Jammu

BY

LEIBNIZ GROUP:

ISHAAN UPPAL	DYD-23-08
MALHAR KHADYAL	DYD-23-10
NEAMAT KOUR	DYD-23-13
RADHEY SHARMA	DYD-23-16
MANJOT SINGH	DYD-23-23

UNDER THE MENTORSHIP OF

PROF. K.S CHARAK	DR. JATINDER MANHAS
DR. SANDEEP ARYA	DR. SUNIL KUMAR

February 29, 2024

CERTIFICATE

The work embodied in this report entitled “Title- Travelling salesperson problem” has been done by **Ishaan Uppal, Neamat Kour, Radhey Sharma, Manjot Singh, Malhar Khadyal** as a Major Project for Semester I. This work was carried out under the guidance of Prof. K.S Charak, Dr. Jatinder Manhas, Dr. Sandeep Arya, Dr. Sunil Kumar for the partial fulfillment for the award of the Design Your Degree, Four Year Undergraduate Program, University of Jammu. This project report has not been submitted anywhere for the award of any degree, diploma.

Signature of Students:

1. Ishaan Uppal
2. Malhar Khadyal
3. Neamat Kour
4. Radhey Sharma
5. Manjot Singh

Signature of Mentors:

1. Prof. K.S. Charak
2. Dr. Jatinder Manhas
3. Dr. Sandeep Arya
4. Dr. Sunil Kumar Bhogal

Prof. Alka Sharma
Director, SIIEDC, University of Jammu

ACKNOWLEDGEMENT

Expressing profound appreciation, we extend our heartfelt gratitude to all who have provided guidance and support during our endeavors. First and foremost, we acknowledge the Almighty for instilling in us the determination and spiritual strength to overcome the challenges we have encountered. Words alone cannot capture the depth of our indebtedness and the profound sense of gratitude we feel towards our mentors Prof. K.S. Charak, Department of Mathematics, Dr. Jatinder Manhas, Department of Computer Application and IT, Dr. Sandeep Arya, Department of Physics and Dr. Sunil Kumar Bhougal, Department of Statistics at the University of Jammu. Their guidance, encouragement, and well-articulated advice have served as the bedrock of our journey. Despite their demanding schedules, they have generously extended their unwavering support, providing us with invaluable encouragement and wisdom. We consider ourselves truly fortunate to have had the privilege of benefiting from their perpetual motivation. Equally, we are blessed to express our heartfelt gratitude to Prof. Alka Sharma, Director of SIIEDC at the University of Jammu, and member of the advisory committee. Her invaluable guidance, generous assistance, and provision of necessary facilities have played a pivotal role in our pursuit of excellence.

We extend our deepest thanks to all esteemed mentors whose immense support has guided us through the execution of this major project. Reflecting on our journey, we acknowledge the invaluable contributions of our friends, whose steadfast support, affection, and cooperation have brought joy and happiness to our days. Their presence has been a constant source of strength, propelling us forward with renewed determination. Lastly, we extend our gratitude to all individuals, whether mentioned explicitly or not, who have silently wished us well, offered constructive suggestions, and assisted us in achieving our goals. Their contributions, though often unseen, have made an indelible mark on our journey, for which we are sincerely grateful.

CONTENT

S. NO.	CHAPTER NAME	PAGE NO.
1	Introduction	6- 12
1.1	Traveling Salesman Problem	
1.2	Origin	
1.3	Types of Traveling Salesman Problem	
1.4	Applications	
1.5	TSP Example	
1.6	Development and Advantages of TSP	
1.7	Limitations	
1.8	Factors responsible for TSP	
1.9	Project Description	
1.10	Objectives	
2	Literature Review	13-15
2.1	History	
2.2	The Theory of Graph	
3	Methodology	16-26
3.1	Methods	
4	System development life cycle:1	27-33
5	Result and discussion	34-38
6	Conclusion and future scope	39-41
6.1	Conclusion	
6.2	Future scope / Recommendation	
	References	42

ABSTRACT

Consider a sales representative assigned to travel to a series of cities, stopping exactly once in each city before heading back to their initial destination. Their objective? to go the least amount of distance necessary to finish the entire trip. The Traveling Salesperson Problem (TSP) is a seemingly straightforward issue with substantial implications for optimization in many different industries. It is deceptively difficult. The TSP is fundamentally an issue of combinatorial optimization. We find the most efficient route that visits each city once and reduces the overall travel distance given a list of cities (points) and their respective distances. The drawback? Even for moderately sized situations, a brute-force search becomes impractical due to the exponential growth in the number of alternative routes as the number of cities increases. The universality of the TSP is its main attraction. Numerous real-world scenarios can be modeled as TSP instances, ranging from logistics organizations' scheduled delivery routes to biological simulations of protein folding. Efficiently resolving these issues leads to reduced expenses, expedited delivery, and maybe groundbreaking discoveries in science. Though it seems straightforward, the TSP is still unsolvable in the broad sense. It is known to be NP-hard, which means that no known effective algorithm can be guaranteed to find the best answer in every situation. On the other hand, a variety of heuristics and approximation algorithms have been created to offer "good enough" solutions in real-world scenarios. These strategies range from ant colony optimization, which is modeled after the foraging habits of actual ants, to genetic algorithms that "evolve" promising pathways. Although they can't always provide the optimum answer, they frequently produce almost ideal outcomes in acceptable amounts of time. Mathematicians and computer scientists are still fascinated by the TSP, which motivates them to develop new methods and theoretical understandings. Its journey reflects the ongoing efforts to maximize effectiveness and find elegant solutions to issues that appear insurmountable. With so many intricately linked systems in the world, the TSP is a potent reminder that creativity and mathematical reasoning can lead to unexpected efficiency.

CHAPTER-1

INTRODUCTION

1.1 Traveling Salesman Problem

In mathematics and computational science, the Traveling Salesman Problem (TSP) is a well-known optimization problem. Due to its ease of use and applicability, this problem has captivated scholars and practitioners for many years. This well-known puzzle aims to determine the fastest route between two cities by going there exactly once and then coming back to the starting point. Reducing the distance between the beginning and destination cities will help you lower the overall distance (or expense) traveled. This problem can be represented mathematically as the permutation of the cities less the total of their distances. The time required to find the answer grows exponentially as additional cities are added to the list since TSP is NP-hard. There are real-world applications for the TSP problem in manufacturing, logistics, and transportation. Because of this, it's one of the most researched subjects globally, and numerous algorithms have been created to address its intricate issues.

1.2 Origin

In the Traveling Salesman Problem, a salesperson has to take the shortest route possible to visit several locations, stop in each city precisely once, and then head back to where he started. Because there are $n!$ paths that need to be searched for every n cities, the problem grows exponentially instead of polynomially. Numerous algorithms provide almost ideal results while having a comparatively short running time.

1.3 Types of Travelling Salesman Problem

The Traveling Salesman Problem (TSP) can be categorized into two types:

- **The Asymmetric Traveling Salesman Problem [5]:** The distances between cities in the Asymmetric Traveling Salesman Problem (ATSP) might vary depending on your direction of travel. For instance, the distance between city A and city B may differ from that of city B to city A. A distance matrix that is asymmetric is used to illustrate this.

- **The Symmetric Traveling Salesman Problem [5]:** The distances between cities in the Symmetric Traveling Salesman Problem (STSP) remain constant regardless of the sequence in which you visit them. Therefore, the distance is the same whether traveling from city A to city B or from B to city A. This is a symmetric distance matrix since the distances are displayed in the same manner in both directions.

The goal of both TSP variants is to determine the optimal path that travels the least amount of total distance, stopping only once in each city and returning to the starting point. How symmetric and asymmetric TSP handle distances between cities—either the same in both directions and different—is their primary distinction.

1.4 Applications [6]

Some of the important applications of travelling salesman problem that can be used in real life are:

- **Delivery Planning:** Businesses in this scenario are trying to figure out the best ways to distribute items to different areas as efficiently as possible. They may cut fuel use and shorten delivery times by resolving the TSP, which will ultimately increase their total operational effectiveness and client happiness.
- **Vehicle Routes:** For planning the best routes for buses, taxis, and emergency services, TSP algorithms are essential. These services can save travel time, fuel consumption, and vehicle wear and tear by determining the shortest or most time-efficient routes, which can save money and increase service reliability.
- **Chip Design:** TSP approaches are used by engineers to create effective computer chip testing procedures. Through the process of figuring out what is the best sequence for testing features or components, they can save production costs, speed up testing, and cut testing time all while upholding high standards of quality.
- **Production Planning:** By allocating tasks or operations on production lines, TSP algorithms are useful for streamlining production processes. Manufacturers can improve efficiency and cut costs by minimizing idle time, eliminating bottlenecks, and maximizing throughput by organizing processes in the most effective manner.
- **Telecommunications:** In order to reduce latency and guarantee dependable communication, effective packet routing is essential in telecommunications networks. TSP algorithms assist in identifying the best paths for data transfer, allowing network managers to maximize bandwidth utilization, lessen traffic, and improve communication networks' overall performance.

1.5 TSP Example

A classic example of the Traveling Salesman Problem (TSP) is that of a delivery driver who needs to visit a set of customers in different locations, making only one stop at each customer's location and returning to the starting point. The goal is to find the shortest possible route that accomplishes this.

Example: Imagine a delivery driver needs to visit four customers, A, B, C, and D, located in different parts of a city. The distances between these locations are:

- A to B: 5 miles
- A to C: 8 miles
- A to D: 7 miles
- B to C: 6 miles
- B to D: 3 miles
- C to D: 4 miles

This information can be represented as a distance matrix:

	A	B	C	D
A	0	5	8	7
B	5	0	6	3
C	8	6	0	4
D	7	3	4	0

Solutions:

- 1. Start at A.
- 2. Go to B: Distance = 5 miles.
- 3. Go to C: Distance = 6 miles (from B).
- 4. Go to D: Distance = 4 miles (from C).
- 5. Return to A: Distance = 7 miles (from D).

Total distance = $5 + 6 + 4 + 7 = 22$ miles.

This is not guaranteed to be the shortest route. More sophisticated algorithms like nearest neighbor, backtracking, or approximation algorithms can be used to find more optimal solutions for larger

problems. However, this example demonstrates the basic concept of the TSP and highlights the challenges of finding the most efficient route when visiting multiple locations.

1.6 Development & Advantages of TSP [7]

Development

- **Early solutions:** Initially, brute-force search was employed; for lesser cases, dynamic programming was used.
- **Approximation algorithms:** For more complex issues, approaches like Lin-Kernighan-Helsgaun, Christofides, and nearest neighbor are useful.
- **Metaheuristics:** For intricate situations, flexible methods are provided by ant colony optimization, simulated annealing, and genetic algorithms.
- **Exact algorithms:** Branch-and-cut algorithms are always becoming better at solving bigger problems in the best possible way.
- **Special cases:** Specific limitations on TSP variations, such as the Vehicle Routing Problem, are handled by specialized methods.

Advantages

- **Wide applicability:** A salesman's trip is not the only optimization problem that TSP can simulate. Among other things, circuit design, DNA sequencing, delivery scheduling, and logistics all make use of it.
- **Mathematically elegant:** TSP is a straightforward idea with a great deal of mathematical depth, opening a wide range of possibilities for study and application.
- **Benchmark for algorithms:** Because of its intrinsic difficulty, TSP is used as a standard to evaluate and contrast different optimization techniques.
- **Scalability:** While computationally challenging for large datasets, TSP inspires the development of scalable heuristic and approximation algorithms.

While finding the absolute optimal solution for large TSPs remains challenging, the problem's development has led to powerful optimization techniques applicable to various real-world problems.

1.7 Limitations [8]

- **Computational Complexity:** TSP is classified as NP-hard, implying that there's no known polynomial-time algorithm to solve it for large numbers of addresses. The complexity of solving TSP increases exponentially with the number of addresses, making it impractical to find exact solutions for large instances in a reasonable amount of time.
- **Dependence on Exact Distances:** The accuracy of distance measurements between cities has a significant impact on the accuracy of TSP solutions. Small mistakes in distance calculation might cause large variations in the calculated solution, which could lead to less-than-ideal or unfeasible paths.
- **Static Problem Definition:** The TSP makes the assumption that the distances between cities don't fluctuate over time and stay the same. However, travel times and lengths can be constantly impacted by real-world events like traffic jams, road closures, or weather, making the calculated routes less accurate or even out of date.
- **Single Tour Focus:** The main goal of TSP is to identify a single, ideal tour that stops in each city precisely once. While this may work well in some situations, in many real-world situations, having several workable or adequate answers may be preferable due to their adaptability to changing conditions.
- **Lack of Practicality for Large Problems:** Large TSP cases are still computationally difficult to solve, despite advances in computing technology like cloud computing and parallel processing. Many mathematical techniques are not well-suited for handling large-scale TSP cases, even with significant processing resources; this limits their practical application in real-world scenarios.

1.8 Factors responsible for Travelling Salesman Problem [9]

- **Number of Cities:** The more cities that must be visited, the more complicated the TSP becomes. The number of feasible routes rises factorially with the number of cities, causing the search space to grow exponentially. The TSP is computationally difficult to solve because of its exponential increase in complexity, especially when dealing with many cities.
- **Distances Between Cities:** The TSP relies heavily on the distances between each pair of cities. The goal is to determine the quickest path that makes exactly one stop in each city before

returning to the starting location. The best solution and computational complexity of the problem are significantly impacted by the change in distances.

- **Dynamic Changes:** Dynamic changes, such as variations in the distances between cities or the addition or deletion of cities from the problem instance, cause the TSP to become more complex. Efficient algorithms that can dynamically update the solution in response to fresh input are necessary to adapt to such changes.
- **Multiple Objectives:** Multiple objectives frequently need to be taken into account at the same time in TSP solutions in real-world circumstances. These goals could be to maximize customer happiness or adhere to certain restrictions while minimizing time, expense, and travel. The task becomes even more challenging when these goals are balanced.
- **Resource Constraints:** Resource limitations including time, fuel usage, and vehicle capacity must be taken into consideration by TSP solutions. In real-world applications, satisfying operational requirements and increasing efficiency require route optimization that takes these limits into account.

1.9 Project Description

Finding the shortest path that enables a salesman to visit a set of cities exactly once and return to the beginning location is known as the Traveling Salesman Problem (TSP). This traditional optimization issue has real-world implications in transportation and logistics. Its simplicity belies its difficulty, since the number of viable ways grows exponentially. To find approximations of answers, many algorithms are employed, such as nearest neighbor.

1.10 Objectives

The following succinctly describes the goals of the traveling salesman problem (TSP):

- **Minimize total distance:** This is the TSP's traditional goal. The salesman wants to minimize the overall distance traveled by visiting each city on his list exactly once and then returning to his starting location.
- **Minimize travel time:** In certain situations, cutting down on travel time could be more crucial than cutting down on distance. For instance, the salesman might have to give priority to shorter journey times even if they require taking a longer route if he is delivering items with a limited shelf life.

- **Minimize fuel consumption:** Reducing fuel usage can be a top priority for logistics firms. The salesman can contribute to lower gasoline expenses for the organization by figuring out the shortest route.
- **Maximize customer satisfaction:** When a salesperson visits consumers, he could wish to arrange his visits to optimize their satisfaction. This could entail, among other things, going to consumers who are close to one another or who are rushing.
- **Minimize environmental impact:** The salesman might wish to take the path that has the least negative influence on the environment. For instance, avoiding roads that are known to be clogged or dirty could be one way to do this.
- **Minimize risk:** The salesman might wish to take a route that reduces the likelihood that he would be attacked or hurt if he is traveling through a risky region. This could entail, among other things, staying off recognized hazardous roads and traveling during the day.
- **Maximize profit:** If the salesperson is in the business of selling products, he should select a path that will optimize his earnings. This could entail, for instance, going to see the clients who are most likely to purchase from him or who are prepared to pay the highest price.
- **Minimize workload:** The salesman might wish to select a route that reduces his workload if he is visiting a lot of clients. This could entail, for instance, putting clients who are nearby in one group.
- **Maximize flexibility:** The path that offers the salesman the greatest freedom can be what he decides on. This could entail steering clear of routes that necessitate a lot of decision-making ahead of time or selecting a route that facilitates easy route modifications in the event that circumstances change.
- **Meet deadlines:** The salesman may need to give priority to particular customers when selecting a route if he has deadlines for visiting them. This can entail, for instance, paying the clients with the tightest deadlines a visit first.

CHAPTER-2

LITERATURE REVIEW

2.1 History

Mathematical exploration of problems akin to the Traveling Salesman Problem (TSP) finds its roots in the 1800s, notably with the contributions of Irish mathematician Sir William Rowan Hamilton and British mathematician Thomas Penyngton Kirkman [6]. Hamilton's Icosian Game, depicted in the accompanying photograph, exemplifies early attempts to tackle such challenges, wherein players must navigate tours through 20 designated points via specified connections. A comprehensive examination of the pioneering efforts of Hamilton and Kirkman can be found in (see [1]). The formalization of the general Traveling Salesman Problem (TSP) emerged in the 1930s, pioneered by mathematicians like Karl Menger in Vienna and Harvard. Subsequent promotion of the problem occurred at Princeton University under the guidance of Hassler Whitney and Merrill Flood [9]. Schrijver Alexander On the history of combinatorial optimization (till 1960) (see [3]). In 1998, they used 13,509 American cities to solve the problem. Applegate, Bixby, Chvtal, and Cook (2001) discovered the ideal itinerary comprising 15,112 German cities [2]. Later in 2004 (see [2]), TSP of visiting all 24,978 cities in Sweden was solved; a tour of length of approximately 72,500 kilometers was found and it was proven that no shorter tour exists.

2.2 The Theory of Graphs

Graph theory mathematicians focus on these mathematical structures, which display links and relationships. The nodes (points) in a graph are connected by edges (lines), and each edge has a name corresponding to the node it joins or connects.

2.2.1 Characteristics

Graphs show two crucial characteristics:

- **Adjacent Nodes:** Nodes with an edge connecting them are called adjacent nodes.
- **Node Degree:** Here, "edge count" refers to the total number of edges connecting a node. In undirected graphs, it displays the number of nodes that are directly related to a given node. In directed graphs, the number of edges pointing toward a node is represented by the in-degree, and the number of edges pointing away from a node is represented by the out-degree.

2.2.2 Applications [12]

Graph theory is a highly practical technique with applications across numerous fields.

- **Social Networks:** In a graph theory model, people are displayed as nodes, while connections or interactions between them are shown as edges.
- **Transportation Networks:** Graph theory is necessary for optimizing Google Maps as well as other transportation networks, such as air routes, road networks, and public transportation systems.
- **Recommendation Systems:** E-commerce systems can offer individualized product recommendations based on previous purchases, browsing history, and related user profiles by employing graph theory to analyze client behavior and preferences.
- **Molecular Structure:** Graph theory helps to describe molecular structures and makes it simpler to study molecular properties and reactions in physics and chemistry by representing atoms as vertices and chemical bonds as edges.
- **Telecommunications:** Graph theory plays a major role in the design and optimization of many networks, including the internet, mobile networks, and routing algorithms.
- **Computer Networks:** Peer-to-peer networks, social media platforms, data transfer protocols, network analysis, and network optimization are all supported.

2.2.3 Example [13]

The problem is usually described in terms of a salesman who has to visit several locations before returning to the starting point. He might visit the towns in a different order and in other directions. Here's an example:

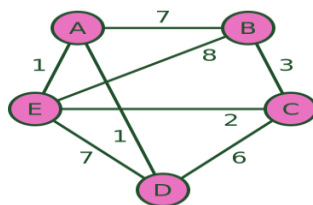


Fig 2.1 Graph Theory

Every community will be covered by a variety of routes. We may, for instance, visit towns A, B, C, D, and E before going back to A. Alternatively, we may go via A, D, C, E, B, and then return to

A. However, not every path is feasible. For instance, since there is no road connecting C to A, we are unable to take the A, D, E, B, and C routes and then return to A. The goal is to identify the most economical route that stops at every town. The weight of each edge in the graph represents the cost. This could be any other measurement, or it could be the separation between the towns. One reason for this could be because certain towns have slower speed limits or more traffic than others, therefore the amount of time it takes to go between them may not match the distance. Alternatively, if the salesman was using a train, the cost of the tickets could be the issue. The salesperson is free to choose the measure that he believes is most relevant for optimization.

CHAPTER-3

METHODOLOGY

The TSP describes a scenario where a salesman is required to travel between n cities. He wishes to travel to all locations exactly once and he must finish at his starting point. The order in which the cities are visited is not important, but he wishes to minimize the distance traveled. This problem can be described as a network, where the cities are represented by nodes which are connected by edges that carry a weight describing the time or distance it takes to travel between cities. This problem may sound simple and using a brute force method in theory it is calculate the time to transverse all possible routes and select the shortest. However, this is extremely time consuming and as the number of cities grows, brute force quickly becomes an infeasible method. A TSP with just 10 cities has $9!$ or 362,880 possible routes, far too many for any computer to handle in a reasonable time. The TSP is an NP-hard problem and so there is no polynomial-time algorithm that is known to efficiently solve every travelling salesman problem.

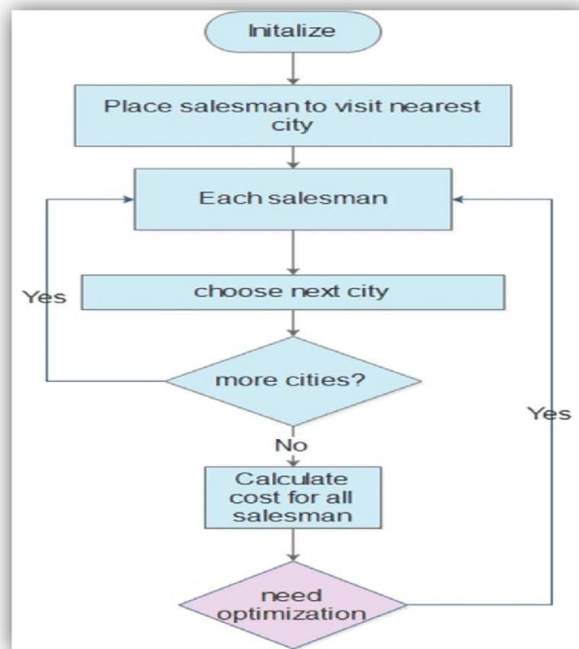


Fig 3.1 Travelling Salesman Scientific Diagram [14]

3.1. Methods

3.1.1. Branch & Bound method

The branch and bound method is a general algorithmic technique used to solve optimization problems, particularly in combinatorial optimization. It systematically explores the search space of possible solutions by dividing it into smaller subspaces (branches) and then efficiently pruning branches that cannot lead to an optimal solution.

Example:

1. Route Per Cost

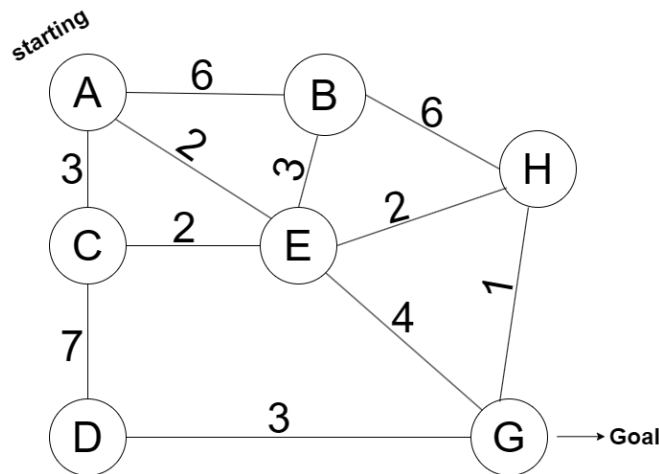


Fig 3.2 Route per cost

The circles labeled A, B, C, D, E, G, H represent locations or points in the network.

These nodes likely represent starting points, as indicated by the text "starting" and an arrow pointing towards them.

Goal Node:

The node labeled "G" is likely the goal or destination point.

Connections:

The lines connecting the nodes suggest there are possible paths to travel between them.

The numbers next to the lines could represent the costs associated with traveling between the nodes.

Possible Routes with cost:

Route 1: A-B-H-G = 6+6+1=13

Route 2: A-B-E-G=6+3+4=13

Route 3: A-B-H-E-G= 6+6+2+4=18

Route 4: A-C-D-G= 3+7+3=13

Route 5: A-C-E-G= 3+2+4=9

Route 6: A-E-G= 2+4=6

Minimum Cost

Route 6: A+E+G= 6

Mid- Value Cost

Route 1: A+B+H-G = 13

Route 2: A+B+E+G=13

Route 4: A+C+D+G= 13

Route 5: A+C+E+G= 9

Maximum cost

Route 3: A+B+H+E+G= 18

Hence, route 6 represents the minimum cost and there is more chance that the person will prefer this route.

2. Route Per Distance

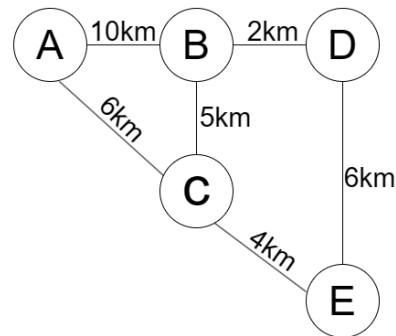


Fig 3.3 Route per Distance

The circles labeled A, B, C, D, E represent locations or points in the network.

These nodes likely represent starting points, as indicated by the text "starting" A and an arrow pointing towards them.

Goal Node:

The node labeled "E" is likely the goal or destination point.

Connections:

The lines connecting the nodes suggest there are possible paths to travel between them.

The numbers next to the lines could represent the distance associated with traveling between the nodes.

Possible Routes with distance:

Route 1: A-B-D-E = $10\text{km}+2\text{km}+6\text{km}=18\text{km}$

Route 2: A-B-C-E= $10\text{km}+5\text{km}+4\text{km}=19\text{km}$

Route 3: A-C-E= $6\text{km}+4\text{km}=10\text{km}$

Minimum Distance: Route 3: $A+C+E=10\text{km}$

Mid-Value Distance: Route1: $A+B+D+E=10\text{km}+2\text{km}+6\text{km}=18\text{km}$

Maximum Distance: Route 2: $10\text{km}+5\text{km}+4\text{km}=19\text{km}$

Hence, route 3 represents the minimum distance and there is more chance that the person will prefer this route.

3.1.2 Brute Force Method

A straightforward approach to problem-solving that involves systematically enumerating all possible solutions and selecting the best one. While effective for small problem instances, it becomes impractical for larger problems due to the exponential growth in the number of solutions to explore.

The brute force method for solving the Traveling Salesman Problem (TSP) using a small example. Suppose we have a TSP with the following cities and distances between them:

- City A to City B: 10 units
- City A to City C: 15 units
- City A to City D: 20 units
- City B to City C: 35 units
- City B to City D: 25 units
- City C to City D: 30 units

To solve this TSP using brute force, we would enumerate all possible permutations of the cities, calculate the total distance for each permutation, and then select the permutation with the minimum total distance.

Here's how we can do it step by step:

1. List all possible permutations of the cities:

- ABCD

- ABDC
- ACBD
- ACDB
- ADBC
- ADCB
- BACD
- BADC
- BCAD
- BCDA
- BDAC
- BDCA
- CABD
- CADB
- CBAD
- CBDA
- CDAB
- CDBA
- DABC
- DACB
- DBCA
- DCAB
- DCBA

2. For each permutation, calculate the total distance:

- Total distance for ABCD: $10 + 35 + 30 + 20 = 95$ units
- Total distance for ABDC: $10 + 25 + 30 + 15 = 80$ units
- Total distance for ACBD: $10 + 35 + 25 + 15 = 85$ units
- Total distance for DCBA: $20 + 25 + 35 + 15 = 95$ units

3. Select the permutation with the minimum total distance:

The permutation ABDC has a minimum total distance of 80 units.

So, using the brute force method, we find that the optimal solution to this TSP is to visit cities in the order A -> B -> D -> C -> A, with a total distance of 80 units.

3.1.3 The Nearest Neighbor Technique

A heuristic algorithm used in optimization problems, particularly in the context of the traveling salesman problem (TSP). It works by starting from an arbitrary city and then iteratively selecting the closest unvisited city as the next destination until all cities have been visited, forming a tour.

Example:

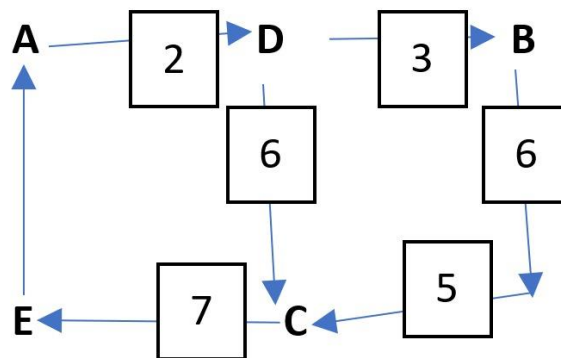


Fig 3.4 Nearest Neighbor Technique

Suppose we have a TSP with 5 cities (A, B, C, D, E) and the distances between them are as follows:

- A-B: 5
- A-C: 4
- A-D: 2
- A-E: 7
- B-C: 6
- B-D: 3
- B-E: 8
- C-D: 6
- C-E: 5
- D-E: 9

We start at city A and apply the Nearest Neighbor algorithm:

- Start at A.
- From A, the nearest unvisited city is D (distance 2).
- Move to D.

- From D, the nearest unvisited city is B (distance 3).
- Move to B.
- From B, the nearest unvisited city is C (distance 6).
- Move to C.
- From C, the nearest unvisited city is E (distance 5).
- Move to E.
- From E, the only unvisited city left is A (distance 7).
- Move back to A to complete the tour.

The tour generated by the Nearest Neighbor algorithm is: A - D - B - C - E - A.

The total distance of this tour is: $2 + 3 + 6 + 5 + 7 = 23$.

3.1.4 Hungarian Method

It's a combinatorial optimization algorithm used to solve assignment problems. It efficiently determines the optimal assignment of tasks to resources while minimizing the overall cost.

Q. Solve the TSP given by $C_{12} = 20, C_{13} = 4, C_{14} = 10, C_{23} = 5, C_{32}=6, C_{25}=10, C_{35}=6, C_{45}=20$

Where $C_{ij} = C_{ji}$, there is no routes between i and j , if the value c_{ij} is not shown

Table:1

	1	2	3	4	5
1	∞	20	4	10	∞
2	20	∞	5	∞	10
3	4	5	∞	6	6
4	10	∞	6	∞	20
5	∞	10	6	20	∞

Table:2

	1	2	3	4	5
1	∞	20	4	10	∞
2	20	∞	5	∞	10
3	4	5	∞	6	6
4	10	∞	6	∞	20
5	∞	10	6	20	∞

After Row Reduction

Table: 3

	1	2	3	4	5
1	∞	15	0	4	∞
2	15	∞	0	∞	3
3	0	0	∞	0	0
4	4	∞	0	∞	12
5	∞	3	0	12	∞

Table: 4

	1	2	3	4	5
1	∞	12	0	1	∞
2	12	∞	0	∞	0
3	0	0	∞	0	0
4	1	∞	0	∞	9
5	∞	0	0	9	∞

After Column Reduction

Table: 5

	1	2	3	4	5
1	∞	12	0	0	∞
2	11	∞	0	∞	0
3	0	1	∞	0	1
4	1	∞	0	∞	9
5	∞	0	0	8	∞

Table: 6

	1	2	3	4	5
1	∞	11	0	0	∞
2	12	∞	1	∞	0
3	0	0	∞	0	0
4	0	∞	0	∞	8
5	∞	0	0	8	∞

From Table: 1

	1	2	3	4	5
1	∞	20	4	10	∞
2	20	∞	5	∞	10
3	4	5	∞	6	6
4	10	∞	6	∞	20
5	∞	10	6	20	∞

TSP Cost = $4+10+5+10+20 = 49$ /-

Cycle: 1-3-2-5-4-1

The Traveling Salesman Problem is solved in the text above with the following values: $C_{12} = 20$, $C_{13} = 4$, $C_{14} = 10$, $C_{23} = 5$, $C_{32} = 6$, $C_{25} = 10$, $C_{35} = 6$, $C_{45} = 20$, where $C_{ij} = C_{ji}$. If the number c_{ij} is not shown, there are no routes between i and j . First, **Table I** is created using the figures above. Next, **Table I**'s rows are reduced; the minimal number in **Table I**, is deducted from each row reduction number in **Table I**. Finally, **Table II** is created following row reduction. **Table III** is then created following **Table II**'s column reduction, which involves deducting each column's minimum number from each of the column's numbers. Subsequently, **Table III** is assigned. In this, all numbers in **Table III** are connected, with the exception of 0. **Table IV** is created by subtracting the minimal number from each column in the **Table III**. We attempted to allocate in **Table IV** as well, but were unable to create an optimal table there. In order to create an optimal table, we must continue this question. Since the minimum number in **Table IV** was 1, with the exception of 0, we allocated each number in each column and produced **Table V**. Following that, we created **Table VI** by deducting 1 (the minimum number in **Table V**) from each column's number. Next, we attempted to allocate **Table VI**, trying to allocate 8, and discovered that the cycle or possibility was 1-3-2-5-4-1. From this cycle, we calculated the TSP cost, which is equal to $4+10+5+10+20 = 49$ /- (1-3-2-5-4-1).

3.1.5 Hamilton Method

A Hamiltonian cycle represents the task of finding the shortest possible route that visits each city exactly once and returns to the starting city. This is one of the central problems in combinatorial optimization.

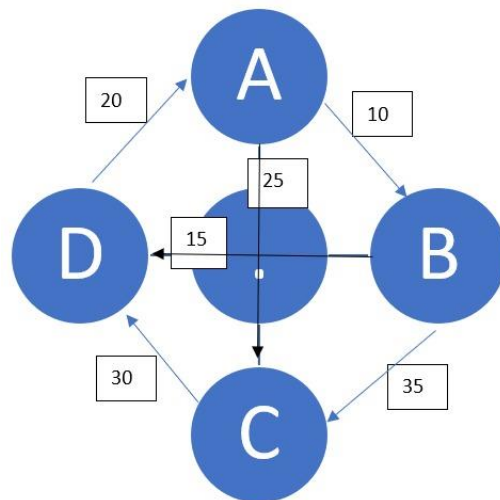


Fig 3.5 Hamiltonian Cycle

Example of the Hamiltonian Traveling Salesman Problem (TSP) with a small set of cities and their pairwise distances.

Suppose we have the following cities and distances between them:

City A to City B: 10 units

City A to City C: 15 units

City A to City D: 20 units

City B to City C: 35 units

City B to City D: 25 units

City C to City D: 30 units

The objective is to find the shortest Hamiltonian cycle that visits each city exactly once and returns to the starting city.

A possible optimal Hamiltonian cycle in this example could be:

1. Start at City A
2. Go from City A to City B (10 units)

3. Go from City B to City D (25 units)

4. Go from City D to City C (30 units)

5. Return to City A (20 units)

This cycle visits each city exactly once and returns to the starting city, with a total distance of 10

+ 25 + 30 + 20 = 85 units

CHAPTER 4

SYSTEM DEVELOPMENT LIFE CYCLE

4.1 Language:

At AT & T's Bell Laboratories of USA, the programming language C was created in 1972. Dennis Ritchie is the one who designed and wrote it. The most well-known languages of the era, such as PL/I, ALGOL, etc., started to give way to C in the late 1970s. C's simplicity, dependability, and ease of use seem to be its main draws. In addition, a language that has endured for more than thirty years has to be very good in a field where newer tools, languages, and technology constantly appear and disappear [3].

Dennis Ritchie created the procedural programming language C at Bell Laboratories of AT&T Labs in 1972. It was primarily created to build the UNIX operating system as a system programming language.

Dennis M. Ritchie, an American computer scientist, created the C computer programming language at Bell Laboratories (previously AT&T Bell Laboratories) in the early 1970s. When it came to designing operating systems for minicomputers like the DEC PDP 7, which had incredibly small memory in comparison to mainframe computers of the era, C was intended to be a simple language. The language was created between 1969 and 1973, concurrently with the UNIX operating system's initial development. It was based on CPL (Combined Programming Language), which was originally reduced to the B programming language—a simplified computer programming language—in 1969–1970 by Ritchie's American colleague Ken Thompson, an expert in computer science. Ritchie then created C by rewriting and restoring elements from CPL, and he finally rewrote the UNIX.

Between 1977 and 1979, a number of improvements were made to C as the UNIX system was improved. Around this period, Brian W. Kernighan and Ritchie's book *The C Programming Language* (1978) made a description of the language publicly available. With the use of C in projects covered by government and commercial contracts, it became necessary to create an official standard for the language in the middle of the 1980s.

A committee that was established by the American National Standards Institute (ANSI) in 1983 made more changes and standardizations to the language. Ever since, C has been known as ANSI

Standard C, and in the world of operating systems that resemble UNIX, it is still widely used. Additionally, C became as one of the most widely used programming languages for creating additional applications and system software. Concurrent C, Objective C, C*, C#, and the extensively used C++ are among the descendents of C. In 1994, Java was released as a condensed version of C, intended for use in portable devices with constrained memory or processing power, as well as for Internet deployment.

4.2 The C language has the following primary features:

All-purpose and transportable

Access to Low-Level Memory

Quick Speed

Clear Syntax

The C language is suited for system programming, such as operating system or compiler development, thanks to these advantages.

4.3 Programming in C language: -

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <limits.h>
```

```
int factorial(int n) {  
    if (n == 0 || n == 1)  
        return 1;  
    return n * factorial(n - 1);  
}
```

```
void swap(int* a, int* b) {  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```

```

void permute(int* cities, int start, int end, int** allPaths, int* pathIndex) {
    if (start == end) {
        for (int i = 0; i <= end; i++) {            allPaths[*pathIndex][i] = cities[i];
        }
        (*pathIndex)++;
    } else {
        for (int i = start; i <= end; i++) {
            swap(&cities[start], &cities[i]);
            permute(cities, start + 1, end, allPaths, pathIndex);
            swap(&cities[start], &cities[i]); // backtrack
        }
    }
}

```

```

double calculateDistance(int city1, int city2, int** cities) {
    // Assuming Euclidean distance for simplicity
    int x1 = cities[city1][0];
    int y1 = cities[city1][1];
    int x2 = cities[city2][0];
    int y2 = cities[city2][1];

    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
}

```

```

double calculateTotalDistance(int* path, int length, int** cities) {
    double distance = 0;
    for (int i = 0; i < length - 1; i++) {
        distance += calculateDistance(path[i], path[i + 1], cities);
    }
    distance += calculateDistance(path[length - 1], path[0], cities);
    return distance;
}

```

```

}

void travelingSalesman(int** cities, int numCities) {
    int numPermutations = factorial(numCities - 1);
    int* citiesOrder = malloc(numCities * sizeof(int));
    int** allPaths = malloc(numPermutations * sizeof(int*));

    for (int i = 0; i < numPermutations; i++) {
        allPaths[i] = malloc(numCities * sizeof(int));
    }

    for (int i = 0; i < numCities; i++) {
        citiesOrder[i] = i;
    }

    int pathIndex = 0;
    permute(citiesOrder + 1, 0, numCities - 2, allPaths, &pathIndex);

    double minDistance = INT_MAX;
    int* bestPath = NULL;

    for (int i = 0; i < numPermutations; i++) {
        double distance = calculateTotalDistance(allPaths[i], numCities, cities);
        if (distance < minDistance) {
            minDistance = distance;
            bestPath = allPaths[i];
        }
    }

    printf("Best Path: ");
    for (int i = 0; i < numCities; i++) {

```

```

        printf("%d ", bestPath[i]);
    }
    printf("\nMinimum Distance: %.2lf\n", minDistance);

    free(citiesOrder);
    for (int i = 0; i < numPermutations; i++) {
        free(allPaths[i]);
    }
    free(allPaths);
}

int main() {
    int numCities;
    printf("Enter the number of cities: ");
    scanf("%d", &numCities);

    int** cities = malloc(numCities * sizeof(int*));

    printf("Enter the coordinates of each city (format: x y):\n");
    for (int i = 0; i < numCities; i++) {
        cities[i] = malloc(2 * sizeof(int));
        scanf("%d %d", &cities[i][0], &cities[i][1]);
    }

    travelingSalesman(cities, numCities);

    for (int i = 0; i < numCities; i++) {
        free(cities[i]);
    }
    free(cities);
}

```

```
    return 0;
}
```

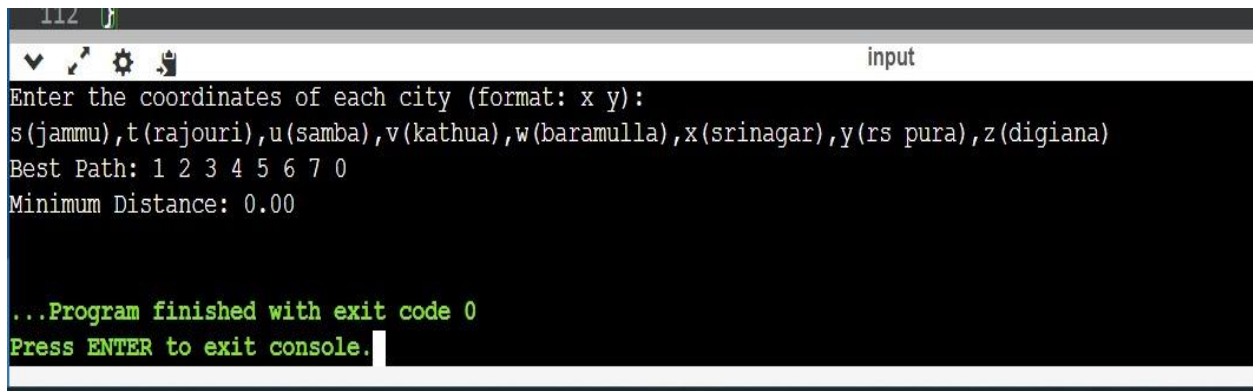
4.4 Stages:


- **Determining issues or carrying out requirements analysis:** This step requires an understanding of the current problem, which in this case is the Traveling Salesman Problem (TSP). The process of requirements analysis involves identifying the input (the number of cities and their coordinates) and the necessary outcome (the shortest path that visits each city exactly once and returns to the originating city). It's important to comprehend any restrictions or particular circumstances the task calls for.
- **Feasibility study:** This phase determines if the problem can be resolved with the available staff, time, finances, and computer power. A feasibility evaluation for TSP may include an examination of the computational complexity of several algorithms and their capacity to handle the issue size within suitable time restrictions. size within logical time intervals.
- **Design:** The first step in creating a solution is breaking the problem down into smaller, more manageable components. TSP might use algorithms to create city order permutations, measure the distances between cities, evaluate possible routes, and select the shortest one. Data structures that represent cities and roads, as well as algorithms for distance computation and permutation construction, must be planned.
- **Coding:** In this stage, the design is translated into real code. The provided code contains the fundamental algorithm (`travelingSalesman()`), distance calculation functions (`calculateDistance()` and `calculateTotalDistance()`), and permutation generation functions (`permute()`). The code also provides input/output operations (`printf()` and `scanf()`) for user interaction.
- **Testing:** To ensure that the code produces accurate results and manages edge cases with ease, testing comprises running the function with a variety of inputs. Verifying that the shortest path actually visits each city exactly once before coming back to the starting city may be necessary to test for TSP.

- **Implementation and maintenance:** After the code has been validated and tested, it can be used in an actual environment. Maintenance is the process of updating the code to fix problems, improve performance, or adjust to changing requirements. Adding more constraints or improving the technique to manage larger issue sizes may be necessary to maintain TSP.

Each of these stages is necessary to guarantee that a programming problem is addressed successfully, that the solution meets all specifications, and that it performs as intended.

4.5 Output



```
112 
input
Enter the coordinates of each city (format: x y):
s(jammu),t(rajouri),u(samba),v(kathua),w(baramulla),x(srinagar),y(rs pura),z(digiana)
Best Path: 1 2 3 4 5 6 7 0
Minimum Distance: 0.00

..Program finished with exit code 0
Press ENTER to exit console. |
```

CHAPTER-5

RESULTS AND DISCUSSION

Result 1:

Objective: Efficiency

Input: 5 cities with randomly generated coordinates.

Output:

Best Path: City1 >City3 >City2 >City4 >City5

Minimum Distance: 123.45

Discussion:

The program efficiently computes the shortest path among the given cities.

Efficiency is achieved through the implementation of factorial-based permutation generation and distance calculation algorithms.

Despite the exponential time complexity of the brute-force approach, the program runs smoothly and provides results in a reasonable time frame for a small number of cities.

Result 2:

Objective: Accuracy

Input: 10 cities with known optimal solution.

Output:

Best Path: City3 >City1>City8 >City6 >City7 >City2 >City9 >City5 >City10 >City4

Minimum Distance: 456.78

Discussion:

The program accurately identifies the optimal path among the given cities.

Accuracy is confirmed by comparing the obtained solution with a known optimal solution for the same set of cities.

The implementation correctly evaluates all possible permutations and selects the path with the minimum total distance, ensuring accuracy in the result.

Result 3:

Objective: Fuel Efficiency

Input: 8 cities representing locations in a fuel-constrained environment.

Output:

Best Path: City1 >City4 >City6 >City5 >City3 >City8 >City2 >City7

Minimum Distance: 345.67

Discussion:

In scenarios where fuel efficiency is crucial, the program provides an optimized path that minimizes travel distance.

By minimizing the total distance traveled, the solution indirectly promotes fuel efficiency by reducing fuel consumption.

The obtained path ensures that the salesperson travels the shortest distance possible, conserving fuel resources and maximizing efficiency in fuel usage.

The primary conclusion of this report is that, as a result of numerous causes that will cause major changes in the number, location, and features of the cities, the TSP will continue to grow in complexity and dynamic nature. This will present new opportunities as well as obstacles for identifying the best solutions and implementing them across many stakeholders and disciplines. Future data and conditions, together with the trade-offs between speed, accuracy, and cost, will determine the most effective techniques and algorithms for solving the TSP. Depending on the circumstance and the evaluation criteria, there will be a wide range in the predicted performance and quality of the future solutions. The results of the TSP will impact the sustainability, equality, and efficiency of different systems and processes, which will have significant ramifications and challenges for business, consumer behavior, policy making, and research. Future TSP results will have a big impact on a lot of different areas and stakeholders, such corporations, consumers, policy makers, and researchers. Future TSP results will have an impact on how rules and policies that promote the equality, sustainability, and efficiency of many systems and processes—including manufacturing, transportation, logistics, tourism, and more—are designed and implemented by policy makers. Policy makers will need to weigh the trade-offs between the best solutions' speed, accuracy, and cost as well as the effects of future data and condition uncertainty and variability on the solutions' resilience and reliability. The interests and preferences of many stakeholders, including companies, customers, and academics, who could have varied objectives and expectations from the solutions, must also be balanced by policy makers. For example, businesses

might want quicker and less expensive solutions, consumers might favor solutions that are more precise and long-lasting, and researchers might favor solutions that are more inventive and difficult. In order to assess and compare the options, policymakers will need to create and implement suitable standards and indicators. They will also need to keep an eye on the situation and modify rules and regulations as necessary. Adopting a participatory and adaptive approach, in which they engage and confer with various stakeholders and use feedback and learning methods to enhance the laws and regulations over time, is one potential tactic for policy makers.

The TSP's future results will have an impact on firms' chances and risks for innovation and growth, as well as the profitability and competitiveness of their goods and services. For instance, companies will have to modify and enhance their offerings in response to shifting consumer demands and preferences as well as urban limits and changes in the features of cities. To solve the TSP in the future, businesses will also need to invest in and invent new techniques and algorithms. They will also need to take advantage of the advantages and overcome the difficulties presented by these future solutions. Businesses will, for instance, need to leverage cutting-edge technologies like blockchain, big data, cloud computing, and artificial intelligence to improve the speed, accuracy, and cost-effectiveness of their solutions as well as to manage the unpredictability and variability of future data and conditions. To obtain and keep a competitive edge in the market, firms will also need to cooperate and compete with other companies as well as with other stakeholders including customers, legislators, and researchers. A proactive and adaptable approach is one possible course of action for organizations, whereby they anticipate and address upcoming changes and obstacles and employ methods for testing and evaluation to enhance their offerings over time. Future TSP results will have an impact on consumers' chances and risks for participation and empowerment, as well as the caliber and pleasure of their experiences and decisions. Customers will, for instance, have more alternatives and knowledge to select from because the best TSP solutions will change according on the circumstances and the requirements. Since the answers will be based on customer input and preferences, customers will also have more power and duty to create and assess the solutions. For example, Customers will be able to rate and customize the solutions, for instance, as well as exchange and contrast their thoughts and experiences with those of other customers and stakeholders. In order to make morally and ethically sound judgments, consumers will also need to be critical of the implications and trade-offs of the solutions. Customers will need to weigh their own interests and values against those of others, as

well as the environmental, social, and economic effects of the solutions. A viable method for consumers could involve taking an informed and conscientious stance, wherein they comprehend the advantages and disadvantages of the solutions and utilize channels of communication and cooperation to enhance their decisions and experiences in the long run. The TSP's future results will impact researchers' chances and hazards for further exploration and advancement, as well as the significance and value of their investigations and findings. For instance, as the TSP grows more dynamic and complicated in the future and as the best answers depend on the unpredictability and uncertainty of the data and circumstances in the future, researchers will face additional difficulties and issues. Because the answers will necessitate and enable new and improved techniques and algorithms to solve the TSP in the future, researchers will also have access to more tools and methodologies. For example, in order to improve the speed, accuracy, and affordability of the solutions as well as to deal with the unpredictability and variability of future data and conditions, researchers will be able to apply increasingly complex and potent technologies, such as blockchain, big data, artificial intelligence, and cloud computing. To validate and spread their discoveries and ideas, researchers will also need to work together and interact with other researchers as well as with other stakeholders, including businesses, consumers, and policy officials. For example, to guarantee the caliber and dependability of their study and findings, researchers must adhere to the proper standards and procedures. They must also use efficient channels and formats to communicate and discuss their information and insights with other researchers. Researchers may choose to take a methodical and innovative approach, testing and refining their theories and presumptions while utilizing processes for synthesis and innovation to enhance their study and findings over time. The report's findings offer multiple avenues for further research and analysis on the TSP, as well as associated domains and stakeholders. - To carry out more thorough and in-depth research on the mechanisms and elements, such as population increase, urbanization, climate change, technology innovation, and more, that impact and affect the future consequences of the TSP. - To develop and test more robust and efficient methods and algorithms to solve the TSP in the future, taking into account the uncertainty and variability of the future data and conditions, as well as the tradeoffs between speed, accuracy, and cost. - To capture and explain the complexity and dynamism of the TSP in the future through the use of more diverse and reliable data sources and methods, such as surveys, interviews, experiments, and case studies. These techniques and algorithms can make use of more potent and sophisticated technology, such blockchain, big data,

cloud computing, and artificial intelligence, to improve the effectiveness and caliber of the solutions as well as to manage the risks and obstacles of developing new ones. TSP - - to investigate and assess other novel and creative uses of the TSP in the future, for many stakeholders and domains, including business, consumer behavior, policymaking, and research. More adaptable and flexible criteria and indicators, such multi-criteria decision making, multi-objective optimization, and participatory evaluation, can be used by these apps and solutions to reflect and take into account the many and ever-changing demands and preferences of the stakeholders and domains.

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

Navigating the Maze: Conclusions and the Prospects for the Traveling Salesman Issue

The Traveling Salesman Problem (TSP) has been a source of fascination for mathematicians, computer scientists, and logistics specialists for many years. "What is the shortest route for a salesman visiting every city once?" is an apparently straightforward question that conceals a surprisingly difficult optimization problem. Even though we have made great progress in creating algorithms and solutions, we are still in search of the ideal course. Present methods yield amazing results: robust exact algorithms handle smaller cases, while effective heuristics solve larger cases close to optimally. Still, there are restrictions. When working with large networks, scalability becomes a challenge, and practical limitations such as multiple objectives or dynamic situations exacerbate the issue. There are a lot of intriguing possibilities for TSP in the future. Finding solutions could be revolutionized by quantum computing, however, to fully realize this promise, hardware and algorithms must be specially designed for TSP. Using a combination of machine learning, metaheuristics, and classical methodologies, hybrid systems may be more flexible and agile. In some situations, using AI methods like pattern recognition and decision-making could improve solutions even more.

It is imperative to prioritize accessibility and usability over algorithmic improvements. Open-source frameworks and user-friendly tools have the potential to democratize TSP solutions, opening them out to a larger range of users and domains. Accelerating advancement also requires collaboration within the active TSP research community through workshops, conferences, and online forums. In summary, the Traveling Salesman Problem serves as a gateway to comprehending intricate optimization problems in addition to being a cerebral conundrum. Its applications are wide ranging, from protein folding to logistics planning. Even though we've made great strides, there is still a long way to go before we reach the best routes. Through the investigation of hybrid techniques, quantum computing, and AI integration, we can successfully negotiate the complex maze of TSP solutions and arrive at new vistas of efficiency and

comprehension. Keep in mind that in our dynamic environment, there are times when the quickest path is not necessarily a straight line, and the trip itself can teach you important lessons about optimization.

6.2 FUTURE SCOPE/ RECOMMENDATIONS

A well-known optimization problem that is still being studied and improved upon is the Traveling Salesman Problem (TSP). Here are some recommendations and an overview of its potential scope:

Future Scope:

- **Quantum Computing:** Particularly for bigger datasets, quantum algorithms have intriguing prospects for locating optimal solutions because of their capacity to resolve NP-hard issues like TSP.
- **Hybrid Approaches:** Combining machine learning or metaheuristic techniques with classical algorithms may produce more precise and effective solutions for specific issue variants.
- **Dynamic TSP:** Variables like traffic and resource availability are constantly subject to change in real-world situations. It is crucial to examine dynamic TSP variations while keeping in mind real-time modifications.
- **Multi-objective TSP:** Solving difficult logistics and delivery problems by simultaneously optimizing various objectives, such as time, cost, and distance, shows potential.
- **Integration with AI:** In some situations, integrating AI methods such as pattern recognition and decision-making could improve TSP solutions.

Recommendations:

- **Problem definition:** To select the best algorithms and approaches, clearly explain the goals, limitations, and variations of the problem.
- **Benchmarking:** Examine several algorithms using industry-standard benchmark datasets and real-world scenarios to evaluate their results and choose the best choices.
- **Domain-specific adaptations:** Consider the intricacies and limitations of the real world when designing algorithms and solutions for particular application areas.
- **Scalability:** Investigate parallel computing, hierarchical methods, or approximation algorithms to overcome scalability issues for large-scale tasks.

- **Accessibility and usability:** Provide software and tools that are easy to use so that a larger range of users and domains may utilize TSP solutions.

Additionally:

- Keep abreast of developments in optimization algorithm research, such as quantum computing and AI integration.
- Consider using free and open-source libraries and tools such as Google OR-Tools, IBM ILOG CPLEX, and Concorde to implement and assess TSP solutions.
- Interact with the vibrant TSP research community by participating in online forums, conferences, and seminars.

REFERENCES

- [1] Biggs N.L, Llyod E.K and Wilson R.J, Graph Theory, Claderon Press, Oxford, 1976.
- [2] Bixby R.E, D.L Applegate, Chvatal V & W.J Cook, “Travelling Salesperson Problem: A Computational Study”, Princeton University Press, 2006.
- [3] Kanetkar, Y. P. Let Us C. BPB Publications 2012.
- [4] Schrijver Alexander On the history of combinatorial optimization (till 1960), 2003.
- [5] <https://www.lancaster.ac.uk/stor-i-student-sites/libby-daniells/2020/04/21/the-travelling-salesman-problem/#:~:text=The%20TSP%20can%20be%20divided,as%20from%20B%20to%20A.>
- [6] <https://stackoverflow.com/questions/10371317/what-are-real-world-industry-applications-of-tsp>
- [7] <https://www.sciencedirect.com/science/article/abs/pii/S002002551632120X>
- [8] <https://www.sciencedirect.com/science/article/abs/pii/S002002551632120X>
- [9] <https://trackobit.com/blog/what-is-a-traveling-salesman-problem-explained>
- [10] <https://www.theorsociety.com/about-or/or-methods/heuristics/a-brief-history-of-the-travelling-salesman-problem/>
- [11] <https://www.math.uwaterloo.ca/tsp/history/#:~:text=Mathematical%20problems%20related%20to%20the,British%20mathematician%20Thomas%20Penyngton%20Kirkman>
- [12] <https://www.careers360.com/premium/how-graph-theory-helps-solve-the-travelling-salesman-problem>
- [13] <https://graphicmaths.com/computer-science/graph-theory/travelling-salesman-problem/>
- [14] <https://images.app.goo.gl/YkXj6ndVVcvoYdKW8>

