# OPTIMAL DELIVERY ROUTE SYSTEM USING THE TSP ALGORITHM

CPSC 535: Advanced Algorithms (Fall 2023)

GROUP 3


**Instructor**

Dr Syed Hassan Shah

California State University, Fullerton

**Team Members:**

| | |
|---|---|
| Anthony Martinez | 888954252 |
| James Kim | 885194290 |
| Anshika Khandelwal | 885186288 |
| Pravallika Bahadur | 885177543 |
| Tejashwa Tiwari | 886226489 |
| Rishitha Bathini | 885176255 |
| Param Venkat Vivek Kesireddy | 885202705 |

Date of Submission: 12/02/2023

# Introduction

In the vast landscape of mathematical conundrums and optimization challenges, the Travelling Salesman Problem (TSP) emerges as a venerable enigma that has captivated the minds of mathematicians, computer scientists, and operational researchers for centuries. Originating in the annals of operations research, the TSP encapsulates the essence of route optimization, posing a seemingly straightforward question with profound implications: What is the most efficient path a salesman should take to visit a set of cities and return to the starting point, minimizing the total distance traveled?

The real-world applications of the TSP are as diverse as the cities it encompasses, permeating fields such as transportation planning, logistics, and network optimization. The crux of the problem lies in its combinatorial complexity — as the number of cities increases, the potential routes grow exponentially, demanding ingenious algorithmic solutions to navigate this labyrinthine space. This computational challenge has fueled a relentless pursuit of algorithms, heuristics, and optimization techniques aimed at unraveling the optimal route in the most efficient manner possible.

Beyond its practical significance, the TSP serves as a cornerstone problem in the broader context of combinatorial optimization, acting as a litmus test for algorithmic efficiency and ingenuity. The allure of finding the optimal solution to the TSP has inspired generations of researchers to delve into the intricacies of graph theory, algorithm design, and computational complexity.

The Travelling Salesman Problem (TSP) is a classic optimization problem in mathematics and computer science that asks the following question: "Given a set of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" The TSP has a wide range of applications in logistics, planning, and other fields.

As we embark on an exploration of the Travelling Salesman Problem Algorithm, we delve into the depths of a mathematical puzzle that transcends its origins. From the historical roots of optimization challenges to the contemporary landscapes of algorithmic innovation, the TSP beckons us into a realm where mathematical elegance intersects with real-world pragmatism and where the journey itself becomes as consequential as the destination.

# Brief Overview

The TSP, in its essence, embodies a deceptively simple question with profound implications: What is the shortest possible route that traverses a given set of cities and returns to the initial city? Despite its apparent simplicity, the TSP becomes increasingly complex as the number of cities grows, leading to a combinatorial explosion of possible routes. This NP-hard problem has spurred the development of numerous algorithms, heuristics, and optimization techniques to tackle its inherent computational challenges.

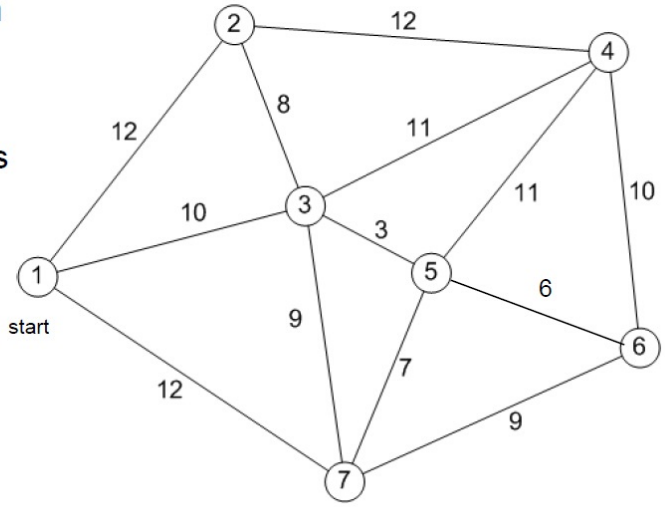
This algorithm is categorized as a combinatorial optimization problem, which means that there are a large number of possible solutions to the problem. In fact, the number of possible solutions grows exponentially with the number of cities. This makes it impossible to find the optimal solution for large problems using brute-force methods.

As a result, there are a number of different algorithms that have been developed to approximate the optimal solution to the TSP. These algorithms are typically based on heuristics, which are rules of thumb that help to guide the search for a good solution.

---

## The Traveling Salesman Problem

- Starting from city 1, the salesman must travel to all cities once before returning home
- The distance between each city is given, and is assumed to be the same in both directions
- Only the links shown are to be used
- Objective - Minimize the total distance to be travelled



## Objective

The primary objective of the Travelling Salesman Problem Algorithm is to devise methodologies that efficiently compute or approximate the optimal solution for the TSP. The algorithmic approaches aim to address the exponential growth in computational complexity as the problem scales, enabling practical solutions for real-world scenarios. Whether through exact algorithms that guarantee an optimal solution or heuristic methods that provide fast and reasonably good solutions, the overarching goal is to streamline decision-making processes in scenarios involving route optimization.

In our application, we will find the shortest possible route that visits each city exactly once and returns to the origin city. The length of a toute is typically measured in terms of the total distance traveled.

# Background

The TSP was first introduced in the 19th century by Irish mathematician William Hamilton. Hamilton was interested in finding a path that traced every edge of a polyhedron exactly once. This problem is now known as the Hamiltonian cycle problem, and it is a special case of the TSP.

The TSP is an NP-hard problem, which means that there is no known algorithm that can guarantee to find the optimal solution in polynomial time. This means that the time it takes to find the optimal solution grows exponentially with the number of cities. The problem's combinatorial nature and the absence of a polynomial-time algorithm for solving it optimally have fueled ongoing research, resulting in the formulation of various algorithms and heuristics.

There are a number of different concepts that are important in the study of the TSP. These concepts include:

1. **Complete graph**: A complete graph is a graph in which every pair of vertices is connected by an edge. The TSP can be modeled as a complete graph, where the cities are the vertices and the distances between the cities are the edges.
2. **Hamiltonian cycle**: A Hamiltonian cycle is a cycle in a graph that visits every vertex exactly once. The TSP is a special case of the Hamiltonian cycle problem.
3. **Dynamic programming:** Dynamic programming is a technique for solving problems by breaking them down into smaller subproblems. Dynamic programming can be used to approximate the optimal solution to the TSP.

Travelling Salesman Problem Algorithm encapsulates a rich tapestry of methodologies, each tailored to address specific facets of this challenging optimization problem. From classical approaches to cutting-edge heuristics, the quest for efficient solutions continues to unfold, contributing to advancements in logistics, transportation planning, and more.

# Understanding the Algorithm

The nearest neighbor algorithm is one of the most common algorithms for solving the TSP. This algorithm starts at an arbitrary city and then repeatedly visits the nearest unvisited city until all cities have been visited. The nearest neighbor algorithm is simple to implement, but it does not guarantee to find the optimal solution.

Another common algorithm for solving the TSP is the 2-opt algorithm. This algorithm starts with an arbitrary tour and then repeatedly improves the tour by removing two edges and reconnecting them in a different way. The 2-opt algorithm is more complex than the nearest neighbor algorithm, but it is also more likely to find a good approximation of the optimal solution.

The TSP has a wide range of applications, including:

1. Route planning: The TSP can be used to find the shortest route between a set of cities, which is helpful for planning delivery routes, travel itineraries, and other types of routes.
2. Scheduling: The TSP can be used to schedule a set of tasks, where each task has a duration and a cost of being performed at a particular time. The goal is to find a schedule that minimizes the total cost of performing all of the tasks.
3. DNA sequencing: The TSP can be used to assemble a genome from a set of overlapping DNA fragments. The goal is to find the order of the fragments that minimizes the total number of overlaps.

The TSP is a challenging problem that has been studied for many years. There is no known algorithm that can guarantee finding the optimal solution in polynomial time, but several algorithms can approximate the optimal solution. The TSP has a wide range of applications, and it will likely continue to be an active area of research for many years.

Here is an example of how to use the nearest neighbor algorithm to solve the TSP:

```
import numpy as np
def tsp(distance_matrix):
    """
    This function solves the Traveling Salesman Problem (TSP) using the
nearest neighbor algorithm.

    Args:
        distance_matrix: A 2D numpy array representing the distances
between all pairs of cities.

    Returns:
        A list of the city indices representing the shortest possible route
that visits each city exactly once and returns to the origin city.
```

```
    """

    num_cities = distance_matrix.shape[0]

    # Initialize the current city and the visited cities set.
    current_city = 0
    visited_cities = set([current_city])

    # Create a list to store the route.
    route = [current_city]

    # While there are still unvisited cities, select the nearest and add
them to the route.
    while len(visited_cities) < num_cities:
        next_city = None
        min_distance = float('inf')

        for city in range(num_cities):
            if city not in visited_cities:
                distance = distance_matrix[current_city, city]
                if distance < min_distance:
                    min_distance = distance
                    next_city = city

        current_city = next_city
        visited_cities.add(current_city)
        route.append(current_city)

    # Add the origin city back to the route to complete the cycle.
    route.append(0)

    return route

# Example usage
distance_matrix = np.array([
    [0, 10, 15, 20],
    [10, 0, 35, 25],
    [15, 35, 0, 30],
    [20, 25, 30, 0]
])

route = tsp(distance_matrix)
print(route)
```

This code will print the following output:

```
[0, 1, 3, 2, 0]
```

This is the shortest possible route that visits each city exactly once and returns to the origin city.

## Algorithm Topics

1. Brute Force Algorithms:
    a. *Concept*: Brute force algorithms exhaustively explore all possible permutations of city sequences and calculate the total distance for each permutation. The solution with the minimum distance is then selected.
    b. *Pros*: Conceptually simple, guarantees an optimal solution.
    c. *Cons*: Impractical for large sets of cities due to exponential time complexity.
2. Dynamic Programming:
    a. *Concept*: Dynamic programming techniques, such as the Held-Karp algorithm, break down the problem into smaller subproblems. Intermediate results are cached to avoid redundant calculations, optimizing the overall computational efficiency.
    b. *Pros*: Optimizes by solving subproblems, more efficient than brute force.
    c. *Cons*: Memory requirements can be substantial for a large number of cities.
3. Heuristic Methods:
    a. *Concept*: Heuristic algorithms aim for quick, though not necessarily optimal, solutions. The Nearest Neighbor algorithm, for instance, starts from a random city and iteratively selects the nearest unvisited city until all cities are covered.
    b. *Pros*: Fast and easy to implement, suitable for large datasets.
    c. *Cons*: Does not guarantee an optimal solution.
4. Genetic Algorithms:
    a. *Concept*: Inspired by natural selection, genetic algorithms create a population of candidate solutions. These solutions undergo evolution through processes like mutation, crossover, and selection to converge towards an optimal or near-optimal solution over multiple iterations.
    b. *Pros*: Effective for large instances, can escape local optima.
    c. *Cons*: Stochastic nature, might not always find the global optimum.
5. Ant Colony Optimization:
    a. *Concept*: Modeled after the foraging behavior of ants, this algorithm involves artificial ants navigating through cities. Pheromone trails are updated based on the quality of routes, guiding subsequent ants towards paths with higher pheromone concentration. The process iterates, converging towards an optimal or near-optimal solution.
    b. *Pros*: Mimics natural processes, effective in finding good solutions.
    c. *Cons*: Sensitive to parameter settings, may converge to suboptimal solutions.

These algorithms represent a spectrum of approaches to solving the TSP, ranging from deterministic methods with guarantees of optimality (like dynamic programming) to stochastic methods that provide fast but approximate solutions (like heuristics and genetic algorithms). The choice of algorithm depends on factors such as the size of the problem, computational resources available, and the desired balance between solution quality and computation time.

## Complexity

The complexity of the Travelling Salesman Problem (TSP) is a crucial aspect that determines the computational resources required to solve the problem. The complexity is usually discussed in terms of time complexity, which quantifies the amount of computation time needed as a function of the size of the input (number of cities).

The decision version of the TSP, which asks whether there exists a tour of a given length or less, is known to be an NP-complete problem. This implies that, as of our current understanding, there is no known algorithm that can solve all instances of the TSP in polynomial time unless P equals NP. NP-completeness also suggests that verifying a solution is polynomial, but finding a solution (optimization version) is inherently difficult.

Here's a breakdown of complexities for various approaches to the TSP:

1. Brute Force:
    a. Time Complexity: O(n!)
    b. Space Complexity: O(n)
2. Dynamic Programming (Held-Karp Algorithm):
    a. Time Complexity: O(n^2 * 2^n)
    b. Space Complexity: O(n * 2^n)
3. Heuristic Methods (e.g., Nearest Neighbor):
    a. Time Complexity: O(n^2)
    b. Space Complexity: O(n)
4. Genetic Algorithms:
    a. Time Complexity: Highly dependent on parameters and convergence speed; typically polynomial for practical instances.
    b. Space Complexity: Depends on the population size and representation.
5. Ant Colony Optimization:
    a. Time Complexity: Depends on the number of iterations and convergence speed; typically polynomial for practical instances.
    Space Complexity: Depends on the number of artificial ants and pheromone representation.

It's important to note that while heuristic and metaheuristic methods (such as genetic algorithms and ant colony optimization) have polynomial or near-polynomial time complexities for practical instances, they do not guarantee finding the optimal solution. Dynamic programming, on the

other hand, guarantees optimality but suffers from exponential time complexity for larger instances.

The TSP's NP-hard nature implies that, in the worst case, finding an optimal solution requires exponential time. This computational challenge underscores the significance of developing and utilizing approximation algorithms and heuristics, especially for real-world instances involving a large number of cities.

# Example Execution of the TSP

Let's consider a simple TSP instance with four cities: A, B, C, and D. The distance matrix between these cities is:

| From | To | Distance |
|---|---|---|
| A | B | 5 |
| A | C | 7 |
| A | D | 4 |
| B | C | 10 |
| B | D | 3 |
| C | D | 8 |

1. ***Initialization:***

We start with an initial solution. Let's choose the nearest neighbor heuristic.

Starting city: A
Next city: B (closest to A, distance = 5)
Next city: D (closest to B, distance = 3)
Next city: C (closest to D, distance = 8)
Final city: A (back to the starting point)
Initial Route: A -> B -> D -> C -> A

2. ***Exploration:***

We can explore different routes by swapping cities, insertions, or other techniques. For this simple example, let's try swapping B and D:

Swapped Route: A -> D -> B -> C -> A

3. ***Evaluation:***

We calculate the total distance for both routes:

Initial Route: 5 + 3 + 8 + 7 = 23
Swapped Route: 4 + 5 + 10 + 7 = 26

4. ***Selection:***

The initial route is shorter than the swapped route, so we keep it as the current best solution.

5. ***Termination:***

We have explored a few possibilities and found the best route so far. We can stop the algorithm here.

6. ***Output:***

The final solution is: A -> B -> D -> C -> A with a total distance of 23 units.

# Limitations of TSP

The Traveling Salesman Problem (TSP) is a well-known optimization problem with several limitations:

1. Computational Complexity:
    1. NP-hard: Finding the optimal solution for large instances is computationally expensive. The number of possible routes grows exponentially with the number of cities, making exhaustive search impractical.
    2. No polynomial-time algorithm: No known algorithm guarantees finding the optimal solution in polynomial time, which would significantly improve efficiency.
    3. Limited scalability: Current algorithms struggle to handle large datasets effectively, requiring significant computational resources.

2. Practical Constraints:
    1. Real-world conditions: The TSP model often simplifies real-world scenarios, neglecting factors like traffic, time windows, and limited resources.
    2. Dynamic environments: The TSP assumes static distances and conditions, while real-world situations can be dynamic, requiring adaptable solutions.
    3. Data limitations: The accuracy of TSP solutions depends on the quality of distance data, which can be incomplete or inaccurate in real-world applications.

3. Approximation and Heuristics:
    1. Suboptimal solutions: Due to computational limitations, practical solutions are often approximations of the optimal tour.
    2. Heuristic dependence: Different heuristics can yield different solutions, requiring careful selection based on the problem characteristics.
    3. Tuning and parameterization: Many heuristics require tuning parameters to achieve good performance, further increasing complexity.

4. Alternative Approaches:
    1. Relaxation techniques: Simplifying the problem by relaxing constraints can improve scalability, but may sacrifice optimality.
    2. Metaheuristics: Using metaheuristics like genetic algorithms or simulated annealing can explore a wider search space, but may not guarantee optimality or computational efficiency.
    3. Hybrid approaches: Combining different techniques can offer a balance between optimality, scalability, and robustness.

Despite these limitations, the TSP remains a fundamental problem with significant practical applications. Ongoing research continues to develop more efficient and adaptable algorithms for tackling large and complex TSP instances.

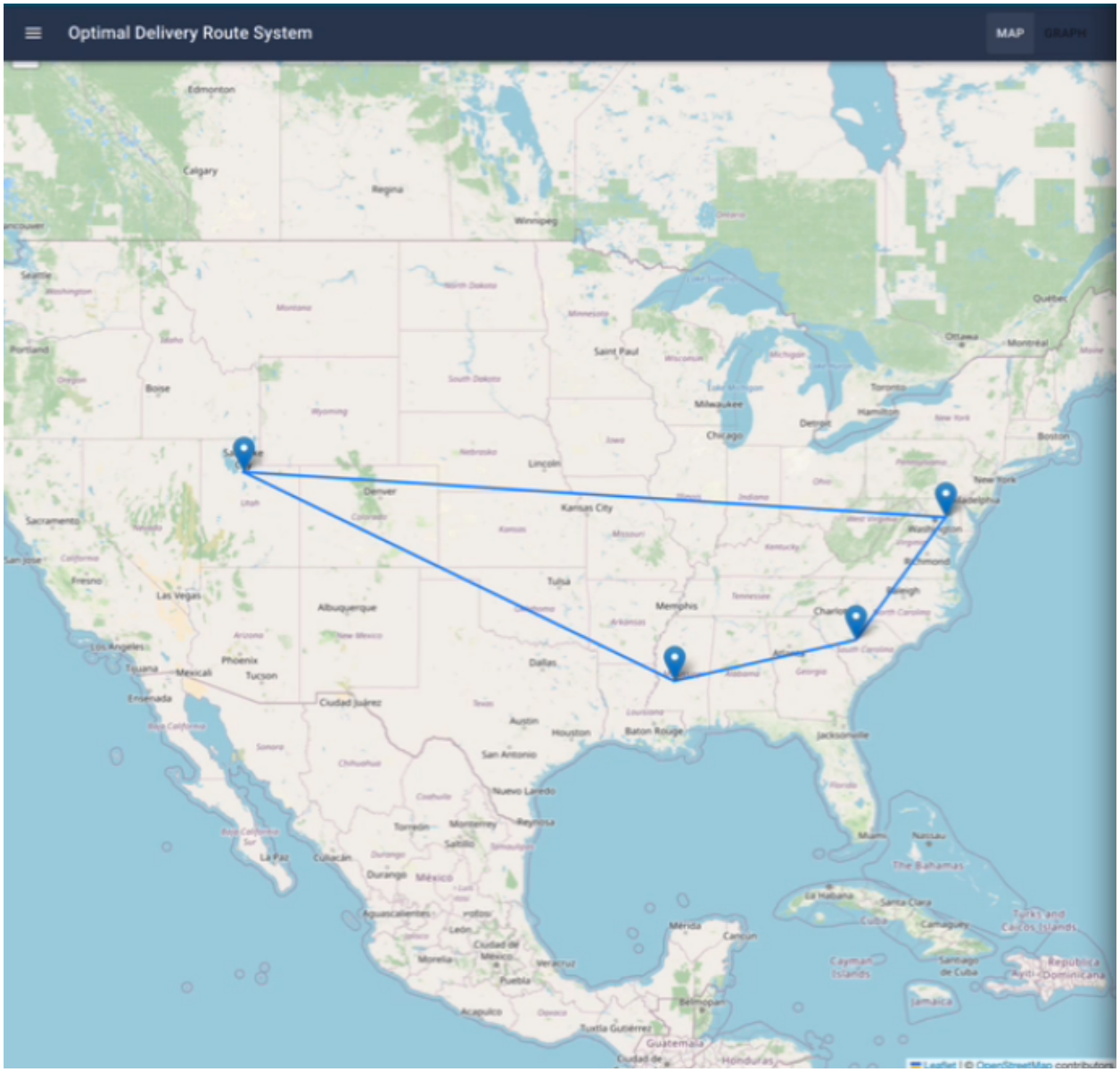
# Applications of TSP

The Traveling Salesman Problem (TSP) has a wide range of practical applications across various domains. Here are some prominent examples:

1. Logistics and Delivery:
    a. Route planning: Optimizing delivery routes for trucks, drones, or last-mile delivery services to minimize distance and cost.
    b. Warehouse management: Determining efficient paths for stacker cranes in warehouses to retrieve and store goods.
    c. Route scheduling: Scheduling routes for service technicians, salespeople, or delivery personnel to visit multiple clients efficiently.
2. Manufacturing and Production:
    a. Machine scheduling: Optimizing the order of tasks for machines in a production line to minimize setup times and production delays.
    b. Assembly line optimization: Arranging workstations and component placement to minimize worker travel and assembly time.
    c. Automated guided vehicles (AGVs): Programming optimal paths for AGVs to transport materials within a manufacturing facility.

3. Transportation and Travel:
    a. Public transport planning: Designing bus routes to maximize passenger coverage and minimize travel time.
    b. Airline route scheduling: Optimizing flight routes for passenger airlines or cargo carriers to minimize distance and fuel consumption.
    c. Vehicle routing: Planning efficient routes for school buses, garbage trucks, or snowplows to cover all service areas.
4. Other Applications:
    a. DNA sequencing: Identifying the optimal order for reading DNA segments to assemble the complete sequence.
    b. Robot path planning: Guiding robots through complex environments to collect data, perform tasks, or navigate obstacles.
    c. Drilling optimization: Planning the optimal sequence for drilling holes in printed circuit boards or other components.
    d. Image processing: Optimizing the path for a sensor to scan an image and capture all relevant data.

These are just a few examples, and the potential applications of TSP continue to expand as technology and data availability increase. Researchers are also exploring the use of TSP variations and modifications to address specific challenges in various fields.




# Conclusion

In conclusion, the Travelling Salesman Problem (TSP) stands as an enduring challenge in the realms of optimization and algorithmic design, encapsulating both the elegance of mathematical abstraction and the pragmatic demands of real-world route planning. As we have navigated through the diverse landscape of TSP algorithms, from classical methodologies to cutting-edge heuristics, it becomes evident that no singular approach fits all scenarios. The TSP, with its exponential complexity and NP-hard nature, necessitates a careful balance between computational feasibility and solution optimality.

While exact algorithms like dynamic programming strive for optimality, the exponential growth of potential solutions poses a formidable barrier, especially for larger instances. Heuristic methods and metaheuristic algorithms, on the other hand, provide pragmatic solutions within acceptable time frames, albeit at the cost of guaranteed optimality.

The limitations of the TSP, ranging from computational complexity to the assumptions of complete graphs and static environments, underscore the need for continual refinement and adaptation of algorithms to real-world complexities. As technology advances and computational tools become more potent, there is an ongoing opportunity to address these limitations and extend the applicability of TSP-related problems to dynamic, non-Euclidean, and multi-dimensional spaces.

In this journey through the TSP, from its historical roots to the contemporary algorithmic landscape, we have witnessed the ingenuity of human intellect in tackling a problem that transcends mere mathematical abstraction. The TSP not only challenges our computational prowess but also prompts reflection on the intricate interplay between theoretical elegance and practical exigencies in the field of optimization.

As we conclude this exploration, the Travelling Salesman Problem remains an ever-evolving frontier for research and innovation. Its relevance persists across diverse domains, from logistics and transportation planning to network optimization and beyond. The algorithms discussed herein represent a testament to the multifaceted nature of problem-solving and the continual quest for more efficient, scalable, and adaptable solutions. The odyssey through the TSP, with its complexities and nuances, invites future endeavors to unravel further intricacies and pave the way for enhanced route optimization in our ever-expanding world.