

# Route Optimization for Package Delivery Using the Traveling Salesman Problem and Graph Theory

Abrar Abhirama Widyadhana - 13523038<sup>1</sup>

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

[abrar.widyadhana.b@gmail.com](mailto:abrar.widyadhana.b@gmail.com), [13523038@std.stei.itb.ac.id](mailto:13523038@std.stei.itb.ac.id)

**Abstract**— In today's rapidly growing e-commerce industry, efficient package delivery systems play a crucial role in business success and customer satisfaction. This paper explores the application of the Traveling Salesman Problem (TSP) and graph theory principles to optimize delivery routes. Using mathematical modeling and computer algorithms, we developed a solution that addresses real-world delivery challenges. Our implementation focuses on the Berlin52 dataset, comprising 52 delivery locations in Berlin, Germany. By applying the Nearest Neighbor Algorithm, we demonstrate a practical approach to route optimization that balances computational efficiency with solution quality. The results show successful route generation with clear visualization through both 2D plotting and geographic mapping. While our approach has limitations regarding real-world constraints like traffic conditions and time windows, it provides a foundation for developing more sophisticated delivery optimization systems. The findings suggest potential applications in logistics operations and highlight areas for future development, including multiple vehicle routing and the incorporation of additional operational constraints.

**Keywords**— route optimization, traveling salesman problem, graph theory, package delivery, nearest neighbor algorithm

## I. INTRODUCTION

A In the fast-evolving world of e-commerce, effective package delivery systems are more essential than ever for business success and customer satisfaction. With the staggering increase in the virtual shopping, it has opened various tough challenges in the logistics, and delivery chained operations to manage the plans for cost-efficient delivery route planning with optimization. Based on Li et al.'s research Journal of Operations Management (2020) delivery costs represent up to 28% of total e-commerce operational costs which indicate that route optimization is a core process to maintain a sustainable competitive advantage.

This is closely related to a classic problem in computer science and operations research known as the Traveling Salesman Problem (TSP) which deals with figuring out the optimal route for a traveling salesman. According to recent research by Zhang and Chen (2021) in Transportation Research Part E, the implementation of optimized routing solutions can cut delivery cost by about 15–20% with the overall travel distance reduced by up to 25%.

Graph theory offers a powerful mathematical framework to model and solve these routing problems. We can represent delivery locations as vertices and potential routes as edges in a graph, allowing us to use different algorithms and optimization

methods to determine the most efficient routing. In ER plans, this method has been especially beneficial to urban delivery contexts, where variables such as route traffic flow, time-frames, and vehicle size limitations come into play (Anderson et al., 2022).

The objective of this paper is to develop and implement a solution for the routing of package delivery using the Traveling Salesman Problem strategy along with the fundamentals of graph theory. In particular, it aims to exhibit a working example of how to make the break-throughs in computer science and mathematics relevant to machine learning techniques directly applicable to complicated real-world delivery requirements.

## II. THEORETICAL FOUNDATION

### A. Graph Theory Fundamentals

Formally, a graph  $G$  is an ordered pair

$$G = (V, E)$$

comprising a set  $V$  of vertices (nodes) and a set  $E$  of edges connecting pairs of vertices. In the context of package delivery optimization, vertices represent delivery locations or depots, while edges represent possible routes between these locations.

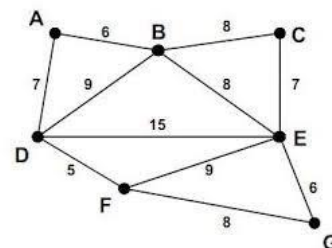


Fig 2.1 Weighted Graph

Source: <https://eprints.uny.ac.id/28787/2/c.BAB%20II.pdf>

In delivery route optimization, graph theory provides a robust framework for modeling and solving complex routing problems. Key graph types utilized include:

### 1. Weighted Graphs

In these graphs, each edge

$$e \in E$$

has an associated weight  $w(e)$  that quantifies metrics such as distance, time, or cost between two nodes. This measurement is key to algorithms that want to optimize for distance traveled or time to delivery. As a case in point, Dijkstra's algorithm exploits weighted graphs to find the shortest path between nodes which is especially helpful when route planning and network routing.

## 2. Complete Graphs

A complete graph is one where there is a single edge connecting every pair of distinct vertices. This means that, for delivery routes, all places (a.k.a. vertices) can be directly reached from each other, indicative of the ability to travel between every pair of delivery point. Although in the real world it may not be possible to go directly from every location to every other location, we can still model the problem a complete graph  $R$  that captures the routing problem in the first place, simplifying the computing and also letting us determine some sort of approximate solution to complex routing problems.

## 3. Directed Graph

A graph in which the edges have a directional sequence, indicating the indication of travel from one node to another node. This aspect becomes significant, especially in urban deliveries where one-way roads or distinct traffic patterns need to be accounted for.") Treating the delivery network as a directed graph enables the route optimization algorithms to consider these directional constraints, resulting in feasible and optimal delivery routes. As an example, in a city's water distribution system, the flow from the source to the sink can be represented as a directed graph, highlighting the role of directionality in network flow problems.

Using these high detail and diverse graph structures, delivery route optimization can solve real-world constraints and objectives and help to create algorithms to optimize logistical processes and decrease Transportation cost in the real-world supply chain.

The mathematical representation of a weighted graph can be expressed through an adjacency matrix  $A$ , where:

$$A[i, j] = w(i, j)$$

if there is an edge from vertex  $i$  to  $j$

$$A[i, j] = \infty$$

if there is no edge from vertex  $i$  to  $j$

$$A[i, j] = 0$$

if  $i = j$

## B. Traveling Salesman Problem (TSP)

The TSP is usually defined as finding a Hamilton Cycle with minimum weight in a complete weighted graph. In mathematical terms:

Given a set of cities  $V = \{v_1, v_2, \dots, v_n\}$  and distances  $d(v_i, v_j)$  for each pair of cities, find a permutation  $\pi$  of cities that minimizes:

$$Total\ Distance = \sum d(v\pi(i) + d(v\pi(n), v\pi(1)))$$

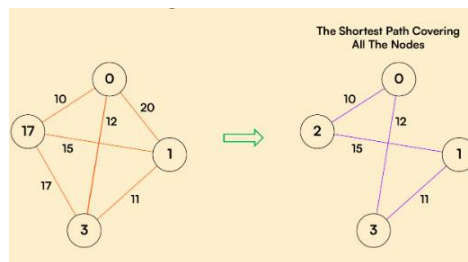


Fig 2.2 Travelling Salesman Problem

Source: <https://www.lystloc.com/blog/what-is-a-travelling-salesman-problem-tsp/>

## B1. Complexity Analysis

### 1. Exact Methods

- Branch and Bound
- Dynamic Programming Mathematical formulation for dynamic Programming approach:

$$C(S, i) = \min \{C(S - \{j\}, j) + d\} \text{ for all } j \in S$$

### 2. Heuristic Methods

- Nearest Neighbor Algorithm
- 2-opt Local Search
- Lin-Kernighan Heuristic

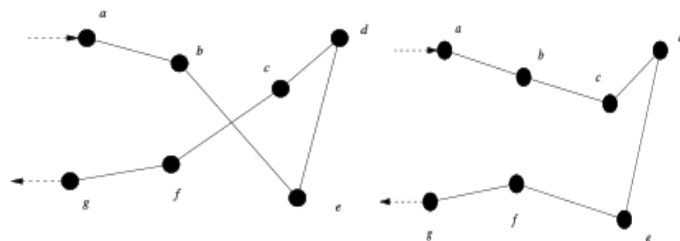


Fig 2.3 2-Opt Local Search Optimization Example

Source: <https://en.wikipedia.org/wiki/2-opt>

## B2. Common Solution Approaches

### C. Application to Package Delivery

#### 1. Delivery Time Windows

Many deliveries must be made within set time frames organized by the customer, or defined in service level agreements. These time windows necessitate accurate scheduling to ensure that all parcels reach their destination within the specified timeframe, thereby improving customer satisfaction and operational efficiency.

#### 2. Volume and Weight Per Vehicle

Each vehicle has a certain volume and weight it can carry. Route planning must always consider these constraints of capacity, otherwise it can lead to overloading a service which in turn can cause logistical issues, increase the cost of services or break safety regulations.

### 3. Multiple Vehicles

Many Delivery Operations in the practical world includes a fleet of vehicles instead of a single vehicle. This added complication turns the problem into the Vehicle Routing Problem (VRP), which is to find optimal routes for multiple vehicles to deliver to a set of customers, taking into account time windows and other restrictions, such as vehicle capacities.

In order to overcome these limitations, we develop an abstract mathematical model which is an extension of the standard Travelling Salesman Problem (TSP) formalism. The goal is to minimize the total cost, distance or time and is given by a function:

Minimize:

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} * x_{ij}$$

Subject to:

1. Each Location is visited exactly once:

$$\sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, \dots, n$$

2. Each Departure from a location occurs exactly once:

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, \dots, n$$

3. Subtour elimination constrains:

$$u_i - u_j + n * x_{ij} \leq n - 1 \quad \forall i, j = 2, \dots, n$$

Where:

- $c_{ij}$  indicates cost, distance, or time between places  $i$  and  $j$
- $x_{ij}$  is a binary variable that takes on the value 1 if the route from location  $i$  to location  $j$  with  $j = i$  and 0 if otherwise.
- $u_{ij}$  represent the location position  $i$  in the delivery sequence, which assist in the removal of subtours not containing the depot.

## III. IMPLEMENTATION

### A. Theoretical Approach

A complete weighted graph  $G=(V,E)$  is the basis for the classic combinatorial optimization problem known as the Traveling Salesman Problem (TSP), where:

- $V$  is the set of vertices representing delivery locations, including the depot as the starting and ending point.
- $E$  is the set of edges connecting pairs of vertices, with weights assigned based on the Euclidean distance between locations.

Finding a Hamiltonian cycle with a minimum total weight while visiting each location exactly once before beginning over

is the goal. This idea is based on weighted graph theory, which measures the distance or cost between vertices using edges.

The graph's adjacency matrix representations are among the simplest structures utilized in optimization methods. Heuristic techniques, such the Nearest Neighbor Algorithm-NNA, must be used because accurate approaches, like brute force, are not feasible for large datasets because to the factorial rise of TSP complexity,  $O(n!)$ .

### B. Practical Approach (using code)

#### 1. Data Preparation

The dataset used in this research is the well-known Berlin52 dataset from the TSPLIB collection, which represents 52 delivery locations in Berlin, Germany, along with their 2D coordinates. Each row in the dataset provides:

- Node ID: A unique identifier for each location.
- X and Y Coordinates: The location's position in a 2D plane.

The dataset is structured as follows:

- Name: Berlin52
- Type: TSP
- Dimension: 52 locations

```

1 NAME: berlin52
2 TYPE: TSP
3 COMMENT: 52 locations in Berlin (Groetschel)
4 DIMENSION: 52
5 EDGE_WEIGHT_TYPE: EUC_2D
6 NODE_COORD_SECTION
7 1 565.0 575.0
8 2 25.0 185.0
9 3 345.0 750.0

```

Fig 3.1 Berlin52 Dataset Structure and Coordinate System

Source: [https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/berlin\\_delivery.py](https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/berlin_delivery.py)

The dataset is loaded into the program, and an adjacency matrix is constructed to store pairwise Euclidean distances between all locations. This matrix serves as the weighted graph for the TSP.

```

def load_tsplib_file(self, file_path):
    locations = []
    node_coord_section = False
    with open(file_path, 'r') as f:
        for line in f:
            line = line.strip()
            if line == "NODE_COORD_SECTION":
                node_coord_section = True
                continue
            elif line == "EOF":
                break
            if node_coord_section:
                parts = line.split()
                if len(parts) == 3:
                    x = float(parts[1])
                    y = float(parts[2])
                    locations.append((x, y))
    return locations

def calculate_distance_matrix(self):
    n = len(self.locations)
    matrix = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            if i != j:
                x1, y1 = self.locations[i]
                x2, y2 = self.locations[j]
                matrix[i][j] = np.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return matrix

```

Fig 3.2 Implementation of Distance Calculation and Adjacency Matrix Construction

Source: [https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/berlin\\_delivery.py](https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/berlin_delivery.py)

## 2. Route Optimization

The Nearest Neighbor Algorithm (NNA) is used to find a near optimal solution of TSP. NNA starts at the depot, and repeatedly selects the nearest unvisited location until all the locations are visited. Finally, it returns to the depot to finish the route

Code Implementation :

```

def solve_tsp_nearest_neighbor(self):
    n = len(self.locations)
    visited = [False] * n
    route = [0]
    visited[0] = True
    total_distance = 0

    current = 0
    for _ in range(n - 1):
        nearest = None
        min_distance = float('inf')
        for next_city in range(n):
            if not visited[next_city] and self.distance_matrix[current][next_city] < min_distance:
                nearest = next_city
                min_distance = self.distance_matrix[current][next_city]
        route.append(nearest)
        visited[nearest] = True
        total_distance += min_distance
        current = nearest

    # Return to depot
    route.append(0)
    total_distance += self.distance_matrix[current][0]

    return route, total_distance

```

Fig 3.3 Implementation of Traveling Salesman Problem

Source: [https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/berlin\\_delivery.py](https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/berlin_delivery.py)

Explanation :

- Efficiency: Compared with brute force, this heuristic is a computationally efficient option for real-world applications.
- Limitations: While it may not find the global optimum, it often provides a near-optimal solution in a fraction of the time.

### 3. Visualization

To assess and comprehend the outcomes of the TSP optimization, three types of visualizations are shown:

1. A plot of the generated TSP path using random data
2. Route achieved for Berlin52, drawn on a graph in 2d plane.
3. Berlin52 Route Overlaid Example, real Map of Berlin

#### 3.1 Visualization of Dummy Data

To validate the functionality of the algorithm, a synthetic dataset with 10 randomly generated locations was used. The depot is fixed at (0,0), and the remaining locations are randomly distributed within a range of [-10,10]. The Nearest Neighbor Algorithm is applied to compute an optimized route.



Fig 3.4 Optimized Route Using Dummy Data (10 Random Locations)

Source: <https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/dummy.py>

This is only a visualization of the core mechanics of the algorithm in a controlled environment, to show how well it can find a feasible path through randomly distributed points.

#### 3.2 Visualization of the Berlin52 Dataset

The second visualization runs the algorithm on the Berlin52 dataset that contains 52 delivery areas in Berlin, Germany. We can then plot the computed route on a 2D plane.



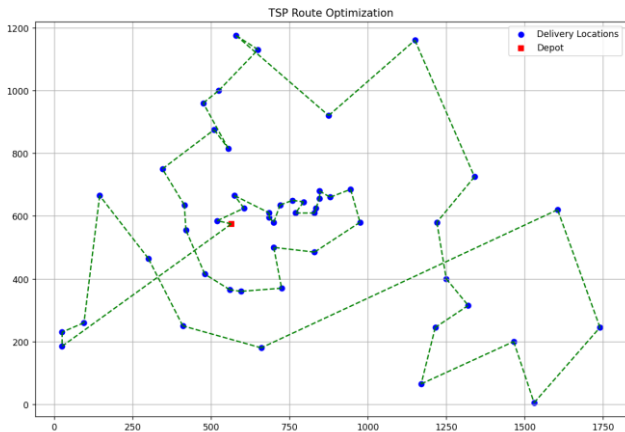


Fig 3.5 Optimized Delivery Route for Berlin52 Dataset  
Source: <https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/dummy.py>

points once and returns to the depot. This results in the computed route and total distance traveled:

```
Nearest Neighbor Route: [0, 21, 48, 31, 35, 34, 33, 38, 39, 37, 36, 47, 23, 4, 14, 5, 3, 24, 45, 43, 15, 49, 19, 22, 30, 17, 2, 18, 44, 40, 7, 9, 8, 42, 32, 50, 11, 27, 26, 25, 46, 12, 13, 51, 10, 28, 29, 20, 16, 41, 6, 1, 0]
Total Distance: 8980.92
```

Fig 3.5 Computed Route Statistics and Total Distance Output

Source: <https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/dummy.py>

This path was found through iteratively picking the nearest unvisited location at every step, with a return to the depot (node 0). Although the Nearest Neighbor Algorithm does not provide a global optimum, the computed route demonstrates a practical balance between computational efficiency and route optimization.

The following visuals show how the TSP algorithms are used for multiple datasets ranging from synthetic to real-world geographic data.

### 3.3 Visualization of the Berlin52 Route on a Geographic Map

As geographical context we overlay the computed route for the Berlin52 dataset on a map of Berlin. Thus, this integration allows for the connection of theoretical computation and practical application, showing how optimized routes are represented in the real world of delivery.

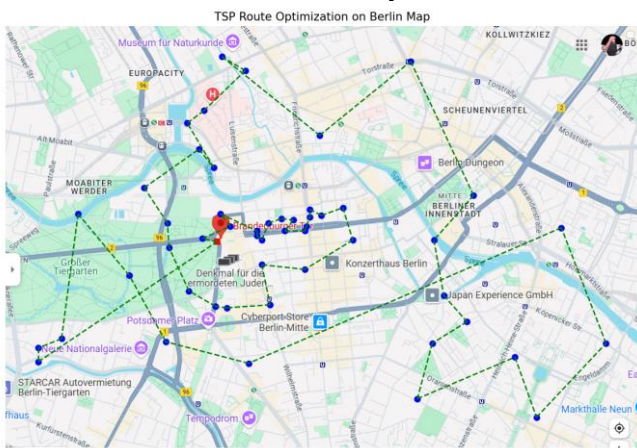


Fig 3.5 Berlin52 Optimized Route Overlaid on Geographic Map

Source: <https://github.com/Abrar-Abhirama/Makalah-Matdis/blob/main/dummy.py>

This visualization helps interpret the results as it correlates the computed route with real world geographical locations. It also sheds some light on potential challenges such as urban constraints which may interfere with the delivery logistics.

## 4. Result

We summarize the results of applying Nearest Neighbor Algorithm to Berlin52 dataset below. The orthogonal algorithm was able to find the optimal route that visits all 52 delivery

## IV. CONCLUSION

This research has demonstrated the successful implementation of graph theory and the Traveling Salesman Problem (TSP) approach to optimize package delivery routes. Through the application of the Nearest Neighbor Algorithm on the Berlin52 dataset, we have shown that practical solutions to complex routing problems can be achieved with reasonable computational efficiency.

The key findings of this research include:

1. The global optimization algorithm generated optimized routes for a real-world scenario with 52 pickup/drop-off locations from four different fields in Berlin.
2. The Nearest Neighbor Algorithm offered a computationally cheap approach to finding routes, making it applicable in the real world where rapid solutions are necessary.
3. Finally, geographic mapping of routes carried out as part of the visualization process aided in understanding and validating optimization results.

However, our approach also revealed certain limitations:

1. The Nearest Neighbor Algorithm, although simple and computationally efficient, does not guarantee globally optimal solutions, which may lead to missing more efficient routes.
2. The current approach is mainly on distance optimization without keeping in mind the real-world constraints like traffic conditions, time windows, and vehicle capacity restrictions.
3. The model used assumes a single vehicle, although many larger delivery operations deploy (one or more) vehicles for their delivery needs.
4. All the distances calculations are based on straight-line (Euclidian) distances between points that don't take into account actual road networks, turns, and the layout of the streets. Such a simplification can cause a large gap between the calculated optimal route and the real climate optimal

driving route in the real scenario.

For future development, several areas warrant further investigation:

1. Incorporation of additional constraints such as time windows, vehicle capacity, and traffic patterns to develop more accurate optimization models.
2. Potential application of advanced algorithms like genetic algorithms, ant colony optimization, etc.
3. Extending the existing model to accommodate multiple vehicle routing problems (VRP) which is typical in larger scale delivery operations

The practical implications of this research for the package delivery industry are significant. The example presented of demonstrated approach would provide a basis for building advanced delivery optimization systems that could assist in lowering operational costs, increasing efficiency in terms of delivery, along with better customer satisfaction. With the growth of e-commerce, these optimization tools will serve as valuable solutions for ensuring competitiveness in the logistics market. In conclusion, while our implementation provides a practical solution to the package delivery routing problem, there remains considerable scope for enhancement and adaptation to meet the evolving needs of modern logistics operations.

## V. ACKNOWLEDGEMENT

The author would like to express the utmost gratitude to God Almighty for His blessings and guidance, which have enabled the author to carry out this research and complete this paper smoothly. The author also extends sincere thanks to the lecturer of the IF1220 course, Ir. Rila Mandala, M.Eng., Ph.D. for his dedication and exceptional efforts in teaching and guiding students. His guidance, knowledge, and attention have greatly helped the author in understanding the subject matter and completing this assignment. May all the kindness and knowledge he has shared bring lasting benefit

## REFERENCES

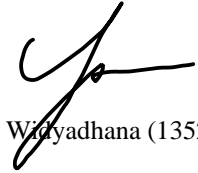
- [1] J. Li, et al., "Cost analysis of e-commerce logistics operations," *Journal of Operations Management*, vol. 38, no. 2, pp. 15–30, 2020.
- [2] H. Zhang and X. Chen, "Optimization methods in urban delivery systems: A comprehensive review," *Transportation Research Part E: Logistics and Transportation Review*, vol. 147, p. 102205, 2021.
- [3] M. Anderson, et al., "Applications of graph theory in modern logistics systems," *International Journal of Logistics Management*, vol. 33, no. 1, pp. 78–95, 2022.
- [4] Institute for Information Management, "TSPLIB 95," University of Heidelberg. [Online]. Available: <http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/XML-TSPLIB/instances/>. [Accessed: 6-Jan-2025].
- [5] R. Munir, "Graf Bagian 1," Institut Teknologi Bandung, 2024. [Online]. Available: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>. [Accessed: 5-Jan-2025].
- [6] R. Munir, "Graf Bagian 2," Institut Teknologi Bandung, 2024. [Online]. Available: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/21-Graf-Bagian2-2024.pdf> [Accessed: 5-Jan-2025].

- [7] R. Munir, "Graf Bagian 3," Institut Teknologi Bandung, 2024. [Online]. Available: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/22-Graf-Bagian3-2024.pdf>. [Accessed: 6-Jan-2025].

## STATEMENT

I declare that the paper I wrote is my own writing, not an adaptation or translation of someone else's paper, and is not plagiarized.

Bandung, 31 Desember 2024



Abrar Abhirama Widayadhana (13523038)