

# Optimal Placement of Waste Bin in ITB Jatinangor Using the p-Median Model in Graph Theory

Arina Azka - 13524049

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jalan Ganesha 10 Bandung

E-mail: [arinaazkaaa@gmail.com](mailto:arinaazkaaa@gmail.com) , [13524049@std.stei.itb.ac.id](mailto:13524049@std.stei.itb.ac.id)

**Abstract**—Current waste bin placement in ITB Jatinangor fails to account for actual students' behaviors during academic and non-academic activities. This paper addresses the issue of optimal placement of a number of waste bins to serve various demand points in the University. To achieve the goal, the problem is modeled using the p-median model in graph theory to reduce the overall weighted distance from each demand point to its nearest facility. In this model, locations are depicted as nodes with each node assigned a demand weight that indicates the level of activity or the volume of waste being produced. Each pair of connected nodes is assigned a value representing the walking distance. To find a solution to the p-median model, this paper utilizes a robust local search algorithm called the Teitz and Bart Algorithm. The implementation is carried out using a Python program that is verified through three scenarios: scattered point distribution, dense clustering, and linear corridors. The results of the tests indicate that the model adeptly adjusts to various geographic positions. In conclusion, the p-median model of trash bin location was resolved by the Teitz and Bart algorithm, which offers an optimal placement.

**Keywords**—graph, p-median, Teitz and Bart algorithm

## I. INTRODUCTION (HEADING 1)

Inefficient waste management in a region remains a lingering concern. As the center of activity for thousands of students, lecturers, and staff, higher-level institutions inherently produce significant volumes of everyday waste dominated by plastic, paper, and food waste from academic activities and daily consumption. It is understood that the waste is often poorly managed due to a lack of necessary materials and facilities, limited labor, and a lack of awareness among students. Institut Teknologi Bandung (ITB) Jatinangor, which has experienced rapid growth in academic activities across its 47-hectare area, is unfortunately no exception to this issue.

According to a study titled “Optimalisasi Pengelolaan Sampah di ITB Jatinangor”, which was the author’s final assignment for the Sustainability course, ITB Jatinangor has the capability to process organic waste into fertilizer. However, there are only four workers assigned to handle waste disposal from 33 temporary shelters to the Integrated Waste Management Installation (IPST). Daily, approximately 1-2 tons of garbage are transported. Since the garbage truck has a capacity of only 400-500 kg, it requires 3-4 trips for complete garbage collection.

On the other hand, students encounter different challenges. One significant issue is the uneven distribution and small size of waste bins, which often leads to the nearest bin overflowing. When one type of waste bin is full, students frequently resort to disposing of waste in bins not designed for it. This practice not only detracts from the aesthetics and cleanliness of the university environment but also complicates the process of separating different types of waste during processing.

It leads to other problems, such as students leaving waste behind at random places. For example, the accumulation of scattered rubbish in the ITB Jatinangor Dormitory canteen even went viral on social media, prompting warnings from various parties.

To address these issues, this paper proposes a shift from an arbitrary waste bin placement approach to a systematic and data-driven solution. The current unoptimized system wastes resources, including labor time spent repeatedly emptying quickly filled bins, and creates inconveniences for the entire academic community. Therefore, a scientific method is essential for efficiently redesigning the facility network..

This paper advocates for implementing the p-median model to determine the optimum placement of waste bin facilities to minimize the travel distance from all demand points to their nearest facilities. The focus is on overall efficiency by considering the locations of waste generation, the weight of demand, and travel distances.

## II. BASIC THEORY

### A. Definition Graph

A graph  $G = (V, E)$  is constructed of a non-empty set of nodes  $V$  and a set of edges  $E$ . Each edge has one or two associated nodes, referred to as its endpoints. An edge is said to connect these endpoints.

### B. Types of Graph

A graph  $G$ 's set of nodes  $V$  may be finite or infinite. However, this paper focuses on finite graphs.

As a discrete structure, there are various types of graphs depending on the types of its edges and the degrees of its nodes. The edges of a graph can be directed and undirected.

An undirected graph is a graph with undirected edges. There are three types of undirected graphs.

- A simple graph is an undirected graph with each edge associated with a pair of nodes without any edges associated with the same pair of nodes.
- A multigraph is an undirected graph that allows multiple edges between the same pair of nodes.
- A pseudograph is an undirected graph with edges connecting a node to itself.

Meanwhile, a directed graph is a graph where each edge has a direction associated with an ordered pair of nodes. There are three types of directed graphs.

- A simple directed graph is a type of directed graph that does not contain any loops or multiple directed edges between the same pair of nodes.
- A directed multigraph is a directed graph that allows multiple directed edges between the same pair of nodes.
- A mixed graph is a directed graph that incorporates both undirected and directed edges.

A directed graph can be obtained by assigning a direction to each edge of an undirected graph. Conversely, an undirected graph can be obtained by eliminating the direction of each edge from its directed graph.

### C. Graph Terminology

#### 1. Adjacent

Two nodes in a graph are considered adjacent if an edge connects them. For example, node 4 is adjacent to node 2 and node 3, but node 4 is not adjacent to node 1.

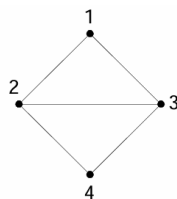


Fig 1. Adjacent, Incidence, and Degree Illustration  
(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

#### 2. Incidency

The incidency of a node  $v$  is the set of all nodes incident to  $v$ . From Figure 1, edge  $(1, 3)$  is incident to node 1 and node 3, but not to node 2 and node 4.

#### 3. Degree

The degree of a node is the number of edges connected to it. From Figure 1, node 1 and node 4 have 2 degrees, while node 2 and node 3 have three degrees.

Meanwhile, in directed graphs, the degree of a node is split into two: the in-degree (number of incoming edges) and the out-degree (number of outgoing edges).

#### 4. Isolated Node

An isolated node is a node that has no edges that are incident on it. Node 5 of Figure 2 is an example of an isolated node.

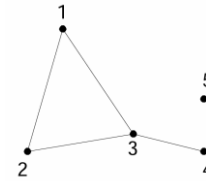


Fig 2. Isolated Node Illustration  
(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

#### 5. Empty Graph

An empty graph is a graph that contains no edges, or each node has a degree of zero.

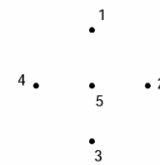


Fig 3. Empty Graph Illustration  
(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

#### 6. Path

A path is a route along the edges of a graph that visits a sequence of nodes, with no nodes and no edges repeated. The length of a path is the number of edges in the path. For example, the path in the graph below is path 0, 4, 8, 9, and 10, with a length of 4.

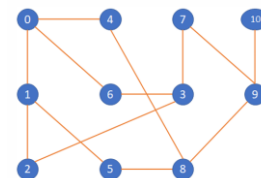


Fig 4. Path and Circuit Illustration  
(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

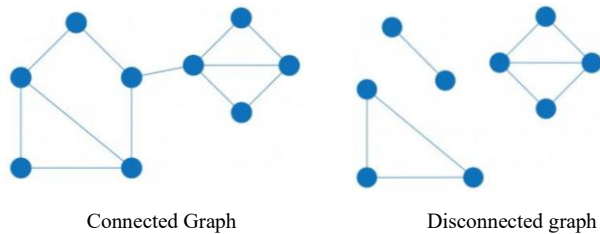
#### 7. Circuit

A circuit is a path that begins and ends in the same node, and no nodes or edges are repeated. In other words, a circuit is a closed path. In Figure 5, a path of 0, 6, 3, 2, 1, 0 is a circuit (say it circuit C).

The length of a circuit is the number of edges in the circuit. The length of the circuit C is 5.

#### 8. Connected

Two nodes are connected if there is a path between them. Graph  $G$  is a graph that is connected if every pair of its nodes is connected. Otherwise,  $G$  is a disconnected graph.



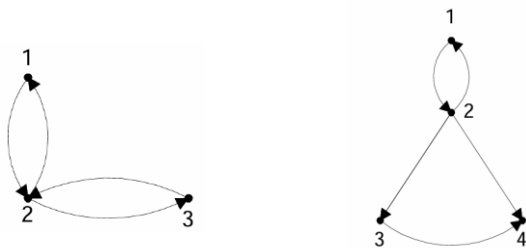
Connected Graph

Disconnected graph

Fig 5. Connected and Disconnected Graphs Illustration

(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

A directed graph  $G$  is considered connected if its corresponding undirected graph is connected. A strongly connected graph is a directed graph with a path for every pair of nodes. Otherwise,  $G$  is a weakly connected graph.



Strongly Connected Graph

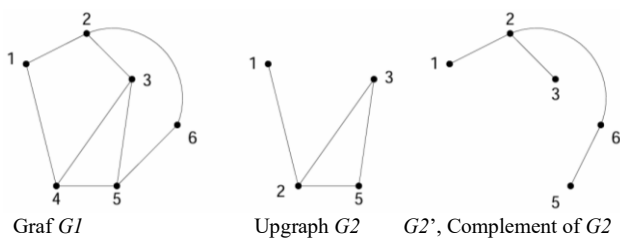
Weakly Connected Graph

Fig 6. Strongly Connected and Weakly Connected Graphs Illustration

(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

## 9. Upgraph and Complement

An upgraph is a graph formed by selecting a subset of its nodes and/or edges. The complement of a graph  $G$  is a graph  $G'$  that has the same set of nodes as  $G$ , but with the opposite edge rules.



Graf  $G1$

Upgraph  $G2$

$G2'$ , Complement of  $G2$

Fig 7. Upgraph and Complement Illustration

(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

A connected component of a graph is a maximal connected subgraph. The components of a graph represent a partition of its nodes into disjoint sets, where every node in a set is reachable from every other node in the same set. A graph with only one component is called a connected graph. The graph below is an example of a graph with four components.

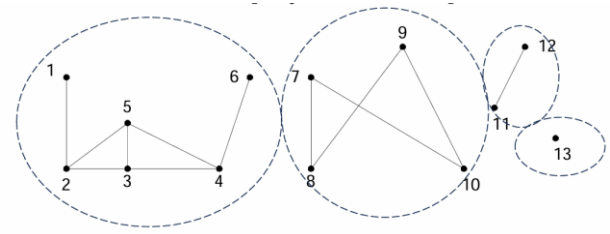
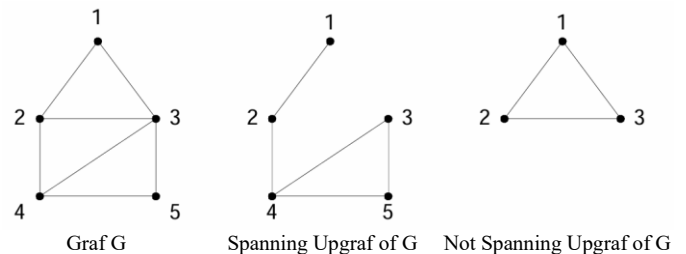


Fig 8. Connected Component Illustration

(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

## 10. Spanning Upgraph

A upgraph  $H$  of a graph  $G$  is a spanning upgraph if its nodes are identical to the set of nodes of the original graph  $G$ .



Graf  $G$

Spanning Upgraf of  $G$

Not Spanning Upgraf of  $G$

Fig 9. Spanning Upgraph Illustration

(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

## 11. Cut-Set

The cut-set of a connected graph  $G$  is the set of edges that, when removed, render  $G$  as a disconnected graph. Therefore, a cut-set of a graph always results in two components. For example,  $\{(1,2), (1,5), (3,5), (3,4)\}$  is a cut set from graph  $G$ .

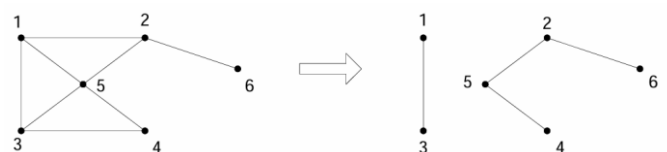
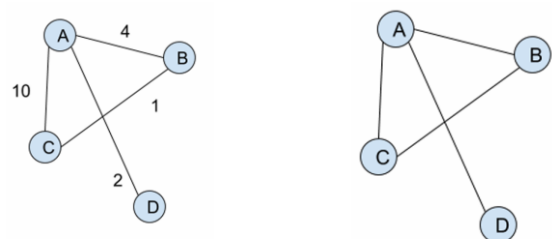


Fig 10. Cut-Set Illustration

(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

## 12. Weighted Graph

A weighted graph is a graph that has each edge assigned a value.



Weighted Graph

Unweighted Graph

Fig 11. Weighted and Unweighted Graphs Illustration

(Source: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>)

#### D. The p-Median Problem

The p-median problem is undeniably one of the most extensively studied models in facility location. Essentially, this problem aims to determine the optimal placement of a specified number of facilities in order to minimize various transportation costs, such as distance or travel time. As a result, demand is allocated to the nearest facility. Thus, the p-median problem can be stated as follows.

“For the given n points with known demand levels, select p of these points to serve as facilities and allocate each demand point to a facility in a manner that minimizes the total weighted distance between the demands and the facilities.”

The term "weighted distance" refers to a value that considers both the distance to a point and its demand weight.

This issue can be addressed through various approaches. Since the initial contributions to the field, numerous techniques have emerged to tackle this problem, as well as adaptations that incorporate additional constraints, cost functions, and assignment policies..

Louis Hakimi was the pioneering researcher who addressed the issue in a network context. In 1964, Hakimi introduced the concept by determining the "absolute median" of a graph. This absolute median extends the idea of the median by allowing the facility to be situated not just at nodes but also anywhere along the edges of the network.

##### Theorem (Hakimi 1964)

An absolute median in a graph is guaranteed to be located at one of its nodes. Hakimi demonstrates that if it is a chosen point on the graph that is not a node, there will always be a node  $v_m$  of G such that

$$\sum_{i=1}^n h_i d_{ij}(v_i, v_m) \leq \sum_{i=1}^n h_i d_{ij}(v_i, x_0)$$

In the following year, Hakimi successfully extended his primary finding (node solution) to encompass the scenario of multiple facilities (say p facilities), called the “p-median” problem. The allocation problem involves assigning demand nodes to their nearest facilities. However, the presence of multiple facilities introduces various possibilities, including the allocation of a demand to more than one facility. For instance, a demand node may be allocated to more than one facility, which might be advantageous in cases if facilities have limited capacity or if customers at demand nodes can choose different facilities on different occasions.

Theorem (Hakimi, 1965): A subset of the node set, denoted as  $V^*_p$ , can be identified that includes p nodes, ensuring that for every collection of p points X within G

$$\sum_{i=1}^n h_i d(v_i, V^*_p) \leq \sum_{i=1}^n h_i d(v_i, X)$$

The theorem demonstrates that at least one optimal solution for the p-median problem involves locating p facilities at the nodes of the network.

#### E. The p-Median Formulation

While Hakimi laid the foundational graph-theoretic concepts, ReVelle and Swain (1970) were the first who formulate the p-median problem as an integer linear program (ILP).

To minimize the overall cost, the p-median problem involves determining the optimal locations for p facilities within a network. The cost is calculated by multiplying the demand at each node i in set I by the distance to the nearest facility. The problem can be expressed using the notation outlined below.

Inputs:

$h_i$ : Demand at node  $i \in I$

$d_{ij}$ : The distance between demand node  $i \in I$  and candidate site  $j \in J$

$p$ : Number of facilities to locate

Decision Variables:

$X_j$ : 1 if we locate at the candidate site  $j \in J$ , 0 if not

$Y_{ij}$ : 1 if the demand weight at node  $i \in I$  is served by a facility at node  $j \in J$ , 0 if not

Using this notation, the p-median problem can be expressed in the following way.

$$\text{Minimize } \sum_{i \in I} \sum_{j \in J} h_i d_{ij} Y_{ij}$$

Constraints:

$$\sum_{j \in J} Y_{ij} = 1 \quad \forall i \in I$$

Each demand point has to be assigned to exactly one facility.

$$\sum_{j \in J} X_j = p$$

There are p facilities that must be placed.

$$Y_{ij} - X_j \leq 0 \quad \forall i \in I; j \in J$$

There is at least one facility that can serve each demand point.

$$X_j \in \{0, 1\} \quad \forall j \in J$$

This constraint states whether the demand point will be maintained.

$$Y_{ij} \in \{0, 1\} \quad \forall i \in I; j \in J$$

This constraint determines whether the area of the  $i$ -th demand point can be served.

The formula above can be interpreted as minimizing the total distance traveled from the node being served ( $i$ ) to the service impulse ( $j$ ) for  $p$  service nodes selected from a number of total nodes to serve a certain number of nodes.

#### F. Teitz and Bart Algorithm

The p-Median method in practice relies on a computer program for support; manually calculating the shortest path and identifying the chosen centers from multiple nodes is not feasible. The p-median problem is classified as NP-hard, meaning there is no known algorithm that has the capability to consistently produce an optimal solution within a time frame that grows polynomially with the size of the problem.

Due to the computational challenges, both researchers and practitioners turn to heuristic approaches. One of the first and most significant of the heuristic algorithms is the node substitution heuristic introduced by Michael Teitz and Polly Bart in 1968, also known as greedy interchange or single vertex substitution.

In this paper, the Teitz and Bart algorithm was selected due to its advantages over alternative methods, such as the Maranzana algorithm, often referred to as the alternating heuristic. The benefits of the Teitz and Bart algorithm include enhanced efficiency and accuracy, as it can yield optimal solutions. This implies that not only does it result in a reduced total distance traveled, but it also provides a better, more stable distribution of demand points. This algorithm examines all nodes to further reduce the total distance traveled. However, a limitation of the Teitz and Bart algorithm is that its analysis time is longer compared to that of the Maranzana algorithm.

The steps taken in the Teitz and Bart algorithm are as follows:

- 1) Select  $p$  nodes as the initial locations of  $p$  facilities.
- 2) Select a served node that is not part of a group of temporarily selected facilities and replace it in each temporarily selected facility. Make the replacement permanent if it results in the most significant reduction in the weighted distance (the replaced facility will no longer be included in the group of temporary selected facilities).
- 3) Perform step 2 for every served node.
- 4) Refer to steps 2 and 3 as a complete cycle. If a complete cycle does not yield any permanent substitutions, cease operations. Otherwise, return to step 2 and initiate a new cycle.

### III. IMPLEMENTATION

#### A. Representing the Problem as A Graph

The basis of this paper is the abstraction of a public place into a graph,  $G = (V, E)$ , whose components are defined as follows.

- Nodes ( $V$ ): A set that represents key locations on campus, ITB Jatinangor, that serve as centers of activity.
- Edges ( $E$ ): A set representing the network of pedestrian paths, sidewalks, and other physical connections that connect pairs of nodes.
- Edge Weight: The shortest walking distance between each pair of connected nodes.
- Demand Weight: Value that represents the level of activity or the volume of waste being produced.

The following components are defined to formalize the problem into the p-median model.

- Demand Point Set ( $I$ ): A set of locations as centers of activities that potentially produce waste.
- Candidate Location Set ( $J$ ): A set of possible locations for waste bin placement.
- Demand Weight( $h_i$ ): Numerical value assigned to each demand point  $i \in I$ , representing the level of waste produced.
- Distance ( $d_{ij}$ ): Distance between each demand point.
- Number of Facilities ( $p$ ): The total count of waste bins that are planned to be installed.

The abstraction process of representing a problem as a graph includes data collection and node identification, demand weighting, and determining the distance between nodes.

##### 1. Data Collection and Node Identification

The graph model is constructed based on spatial data and facility information from public documents, including the official map of ITB Jatinangor provided by the university, satellite images, and directories of buildings and rooms. From these references, significant locations were pinpointed as activity centers in ITB Jatinangor.

##### 2. Demand Weighting

In an ideal scenario, the demand weight ( $h_i$ ) for every node  $i$  relies on empirical data about the waste generated at that specific location. However, due to the lack of such detailed data, a proxy approach is created to estimate the level of activity or "foot traffic" on a scale of 1-100 at each node based on the crowds on a daily basis.

##### 3. Determining Distance Between Nodes

The distances between a pair of nodes are determined through Geographic Information System (GIS) and application



programming interfaces (APIs) from mapping platforms. By mapping the geographic coordinates of each node, the distances are computed along the campus trail network, ensuring that the values accurately reflect the minimal walking distances that pedestrians would cover. The determined distances are then represented as a  $p \times p$  matrix.

## B. Implementing Algorithm

To transform the theoretical model into a functional decision-making instrument, this paper developed a program using Python. This program implements the p-median model using the Teitz and Bart algorithm. However, the program was only created using dummy data as a general representation of the problem.

### 1. Initializations

The initial phase of implementation includes established libraries to facilitate the functionality of the program. The use of libraries in the program is described as follows.

- pandas: To manage data in tabular format. In this program, the data is called a DataFrame.
- numpy: To handle numerical calculations and matrix operations.
- matplotlib: To take data and calculation results, then turn them into informative visual graphs.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Fig 12. Libraries Usage  
(Source: Author's Source Code)

This step also involves setting up the PmedianSolver class, which imports all the necessary data and encompasses all required placement functionality.

The class constructor takes in a DataFrame that includes a node list, demand weights, and a distance matrix through the `__init__` method.

```
def __init__(self, dist_matrix, weights, p):
    self.dist_matrix = dist_matrix
    self.weights = weights
    self.p = p
    self.num_locations = dist_matrix.shape[0]
```

Fig 13. Code of `__init__`  
(Source: Author's Source Code)

Besides that, the class constructor is also responsible for calculating the total cost for each node through the `_calculate_total_cost` method.

```
def _calculate_total_cost(self, open_indices):
    total_cost = 0
    assignments = {idx: [] for idx in open_indices}

    for i in range(self.num_locations):
        distances_to_open = self.dist_matrix[i, open_indices]
        min_dist_idx_local = np.argmin(distances_to_open)
        min_dist = distances_to_open[min_dist_idx_local]

        assigned_facility_idx = open_indices[min_dist_idx_local]

        total_cost += self.weights[i] * min_dist
        assignments[assigned_facility_idx].append(i)

    return total_cost, assignments
```

Fig 14. Code of `_calculate_total_cost`  
(Source: Author's Source Code)

### 2. Implementation of Teitz and Bart Algorithm

The core of the program revolves around the implementation of Teitz and Bart Algorithm in the PmedianSolver class. The procedure operates as follows.

- 1) Begin by randomly generating an initial solution by choosing  $p$  locations from a pool of candidate nodes.
- 2) The algorithm seeks to replace each “open” facility with every closed facility.
- 3) For each proposed exchange, the overall cost (calculated as the sum of weighted distances) is reassessed using the imported distance matrix. If the exchange leads to a reduction in cost and is accepted, then the collection of open facilities is modified accordingly.
- 4) The sequence of steps 1-3 continues until no single exchange can yield a decrease in the total cost. At this stage, the solution is deemed a local optimum.

The algorithm is implemented as follows.

```
def TeitzBartAlgorithm(self):
    current_placement = np.random.choice(range(self.num_locations), self.p, replace=False).tolist()
    best_cost, _ = self._calculate_total_cost(current_placement)

    improved = True
    while improved:
        improved = False
        closed_locations = [j for j in range(self.num_locations) if j not in current_placement]

        for open_idx in current_placement:
            for closed_idx in closed_locations:
                temp_solution = [idx for idx in current_placement if idx != open_idx]
                temp_solution.append(closed_idx)

                temp_cost, _ = self._calculate_total_cost(temp_solution)

                if temp_cost < best_cost:
                    best_cost = temp_cost
                    current_placement = temp_solution
                    improved = True
                    break
            if improved:
                break

    final_cost, final_placements = self._calculate_total_cost(current_placement)
    return current_placement, final_cost, final_placements
```

Fig 15. Code of Teitz and Bart Algorithm Implementation  
(Source: Author's Source Code)

### 3. Visualization

Although the PmedianSolver does not rely on coordinates, visualization of the result still requires spatial data to plot an interpretable map. Therefore, the program includes a visualization function that uses the matplotlib library to draw a

graphic representation of the Cartesian coordinates to show the plotting result. The visualization function is as follows.

```
def Visualization(locations_df, open_indices, assignments):
    demand_points = locations_df[['X_Coordinate', 'Y_Coordinate']].values

    plt.figure(figsize=(12, 10))

    plt.scatter(demand_points[:, 0], demand_points[:, 1], c='red', label='Demand Point')

    for i, row in locations_df.iterrows():
        plt.text(row['X_Coordinate'] + 0.5, row['Y_Coordinate'] + 0.5, f'{row["Location_ID"]}: ({i:(row["Demand_Weight"])}')

    open_coords = demand_points[open_indices]
    plt.scatter(open_coords[:, 0], open_coords[:, 1], c='green', marker='*', s=250, label='Optimal Waste Bin Location', zorder=5)

    for facility_idx, assigned_points in assignments.items():
        facility_coordinate = demand_points[facility_idx]
        for point_idx in assigned_points:
            point_coord = demand_points[point_idx]

            plt.plot([facility_coordinate[0], point_coord[0]], [facility_coordinate[1], point_coord[1]], 'g--', alpha=0.5)

    plt.title('Visualization of Optimal Placement')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.grid(True)
    plt.show()
```

Fig 16. Code of Visualization Function  
(Source: Author's Source Code)

#### 4. Main Program

As the handling for the overall program, the main section starts with the preparation of data, where raw details about locations, names, coordinates, and request weights are defined and subsequently organized into a structured tabular format utilizing a pandas DataFrame. The distance matrix should be calculated using GIS, but in this paper, the main program prepares the distance matrix by calculating the linear distance between two nodes.

After the data is organized, the program creates a PMedianSolver object by supplying the distance matrix, weights, and the number of facilities  $p$  that should be established. Following the completion of the computation, the main program receives the selected locations, the minimum total cost, and allocation information, then presents them to the terminal in a clear and informative format for the user.

```
if __name__ == '__main__':
    data = {
        'Location_ID': ['01', '02', '03', '04', '05', '06', '07', '08'],
        'Location_Name': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'],
        'X_Coordinate': [10, 85, 45, 23, 90, 65, 50, 35],
        'Y_Coordinate': [20, 78, 55, 60, 40, 15, 80, 5],
        'Demand_Weight': [15, 8, 12, 10, 7, 9, 11, 13]
    }
    site_df = pd.DataFrame(data)

    p = 2

    weights = site_df['Demand_Weight'].values

    coords = site_df[['X_Coordinate', 'Y_Coordinate']].values
    num_locs = len(coords)
    dist_matrix = np.zeros((num_locs, num_locs))
    for i in range(num_locs):
        for j in range(num_locs):
            dist_matrix[i, j] = np.linalg.norm(coords[i] - coords[j])

    solver = PMedianSolver(dist_matrix, weights, p)
    optimal_indices, total_cost, assignments = solver.TeitzBartAlgorithm()

    print(f"Optimal Placement of {p} Waste Bins")
    print(f"~" * 50)
    print(f"Total Cost: {total_cost:.2f}")

    print("\nSelected Locations:")
    optimal_locations = site_df.loc[optimal_indices]
    for _, row in optimal_locations.iterrows():
        print(f"- {row['Location_ID']}: {row['Location_Name']}")

    print("\nDemand Points Allocations to the Assigned Waste Bins")
    for facility_idx, assigned_points_indices in assignments.items():
        facility_name = site_df.loc[facility_idx, 'Location_Name']
        assigned_names = site_df.loc[assigned_points_indices, 'Location_Name'].tolist()
        print(f"\nWaste bin in '{facility_name}' is assigned to:")
        for name in assigned_names:
            print(f"  - {name}")

    Visualization(site_df, optimal_indices, assignments)
```

Fig 17. Code of Main Function  
(Source: Author's Source Code)

#### IV. TESTING

A series of tests was conducted using three different datasets to assess the performance of the implemented p-median solver program. Each dataset was designed to represent different geographic distribution scenarios: a scattered (Scenario 1), a grouped (Scenario 2), and a linear (Scenario 3).

```
data = {
    'Location_ID': ['01', '02', '03', '04', '05', '06', '07', '08'],
    'Location_Name': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'],
    'X_Coordinate': [10, 85, 45, 23, 90, 65, 50, 35],
    'Y_Coordinate': [20, 78, 55, 60, 40, 15, 80, 5],
    'Demand_Weight': [15, 8, 12, 10, 7, 9, 11, 13]
}
site_df = pd.DataFrame(data)

p = 2
```

Fig 18. Dataset of Scenario 1  
(Source: Author's Source Code)

Scenario 1 tests the program's ability to place 2 waste bins in 8 randomly distributed demand points. The distribution has no clear cluster pattern, so it challenges the program to find the most efficient “center of gravity”.

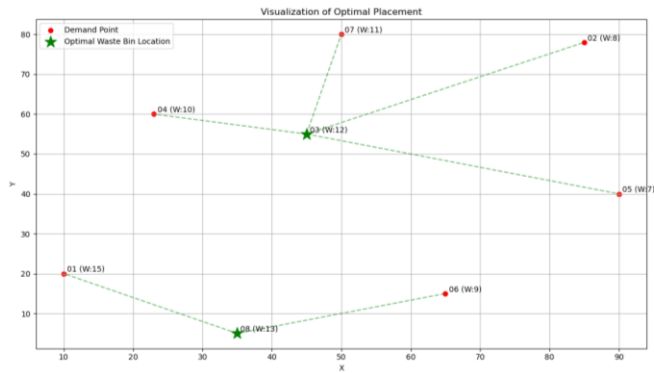


Fig 19. Visualization of Scenario 1  
(Source: Author's Source Code)

```
Optimal Placement of 2 Waste Bins
-----
Total Cost: 1929.15

Selected Locations:
- 03: C
- 08: H

Demand Points Allocations to the Assigned Waste Bins

Waste bin in 'C' is assigned to:
- B
- C
- D
- E
- G

Waste bin in 'H' is assigned to:
- A
- F
- H
```

Fig 20. Result of Scenario 1  
(Source: Author's Source Code)

As seen in the visualization and results above, the program successfully partitions the area into two distinct service zones. Location 08 (H), which carries a high-demand weight (13), serves as the hub for the lower cluster, catering to points A, F, and H. On the other hand, location 03 (C), which also has a significant demand weight (12), serves effectively as the hub for the more scattered points in the upper and right regions (B, C, D, E, G). The results demonstrate the model's capability to identify and form efficient service clusters in the site with a lack of natural grouping and strategically position the facilities in locations that reduce weighted distances to the most advantageous groups of points.

```
data = {
  'Location_ID': ['01', '02', '03', '04', '05', '06', '07', '08', '09',
    '10', '11', '12'],
  'Location_Name': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
    'L'],
  'X_Coordinate': [10, 15, 20, 12, 25, 88, 95, 18, 90, 22, 14, 28],
  'Y_Coordinate': [15, 25, 10, 20, 28, 80, 85, 18, 92, 12, 22, 14],
  'Demand_Weight': [20, 25, 18, 15, 22, 7, 5, 12, 4, 16, 28, 19]
}
site_df = pd.DataFrame(data)

p = 3
```

Fig 21. Dataset of Scenario 2  
(Source: Author's Source Code)

Scenario 2 is designed to test the program on placing 3 waste bins in 12 uniformly distributed demand points with two distinct clusters. This scenario aims to evaluate how the program distributes facilities across areas with different densities and demands.

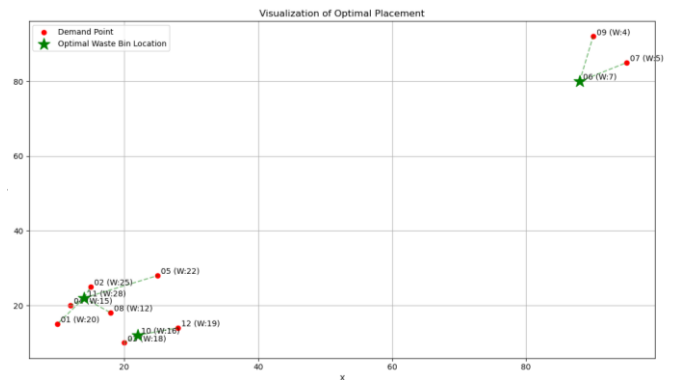


Fig 22. Visualization of Scenario 2  
(Source: Author's Source Code)

```
Optimal Placement of 3 Waste Bins
-----
Total Cost: 889.02

Selected Locations:
- 11: K
- 06: F
- 10: J

Demand Points Allocations to the Assigned Waste Bins

Waste bin in 'K' is assigned to:
- A
- B
- D
- E
- H
- K

Waste bin in 'F' is assigned to:
- F
- G
- I

Waste bin in 'J' is assigned to:
- C
- J
- L
```

Fig 23. Result of Scenario 2  
(Source: Author's Source Code)

The program positions two of the three facilities, 10 (J) and 11 (K), directly within the dense cluster located at the bottom left. This choice is effective since this area contains numerous nodes with high demand weights (for example, K with a weight of 28, B with 25, and E with 22) situated close to one another. By placing two facilities in this area, the internal travel distance within the dense cluster can be significantly reduced. The third facility, 06 (F), is wisely situated to cater to the outlying cluster at the top right (F, G, I). This demonstrates that the model not only places facilities at the center of mass but is also capable of allocating the number of facilities in relation to the density and overall demand weights in a specific area.

```
data = {
  'Location_ID': ['01', '02', '03', '04', '05', '06', '07', '08', '09',
    '10', '11', '12', '13', '14', '15'],
  'Location_Name': ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
    'L', 'M', 'N', 'O'],
  'X_Coordinate': [5, 10, 15, 22, 30, 38, 45, 55, 63, 70, 78, 85, 90, 95,
    100],
  'Y_Coordinate': [8, 15, 12, 20, 25, 33, 40, 45, 55, 60, 68, 72, 80, 85,
    90],
  'Demand_Weight': [22, 18, 15, 10, 8, 5, 4, 6, 9, 11, 14, 17, 20, 25, 28]
}
site_df = pd.DataFrame(data)

p = 5
```

Fig 24. Dataset of Scenario 3  
(Source: Author's Source Code)



Last, the third scenario evaluates the model using 15 demand points set in a linear arrangement as pertinent for positioning facilities along highways, pipelines, or transportation routes. With  $p = 5$ , the objective is to determine whether the model can distribute facilities uniformly along the corridor.

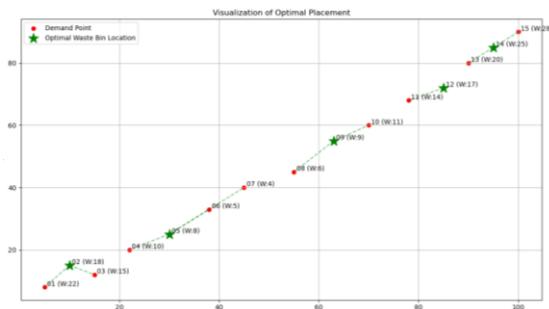


Fig 25. Visualization of Scenario 3  
(Source: Author's Source Code)

```
Optimal Placement of 5 Waste Bins
-----
Total Cost: 1136.22

Selected Locations:
- 02: B
- 14: N
- 12: L
- 05: E
- 09: I

Demand Points Allocations to the Assigned Waste Bins

Waste bin in 'B' is assigned to:
- A
- B
- C

Waste bin in 'N' is assigned to:
- M
- N
- O

Waste bin in 'L' is assigned to:
- K
- L

Waste bin in 'E' is assigned to:
- D
- E
- F
- G

Waste bin in 'I' is assigned to:
- H
- I
- J
```

Fig 26. Result of Scenario 3  
(Source: Author's Source Code)

Based on the visualization for this scenario, the five facilities are placed at relatively regular intervals along the diagonal corridor. Each selected facility becomes the center for a local segment along the corridor. For example, 02 (B) serves the lower left end, 05 (E) serves the next section, and so on up to 14 (N), which serves the upper right end. This placement is intuitively the most efficient way to minimize the total travel distance in a linear configuration. This proves that the  $p$ -median algorithm is very effective for the problem of route planning and facility placement along a given path.

In summary, these three testing scenarios thoroughly confirm that this version of the  $p$ -median program is resilient, adaptable, and proficient in generating rational and quantitatively effective placement solutions across different topological configurations and demand patterns.

## V. CONCLUSION

This paper proved the efficacy of the  $p$ -median model in graph theory in determining the optimal waste bin placement. The real-world problem is represented as a weighted graph where the potential waste-producing location stands as a node, and the distances between those locations are represented by edges. To determine the optimal placement, a Python program is developed by using the Teitz and Bart Algorithm to select the ideal location and visualize spatial data in the Cartesian coordinates.

Through testing of three data sets, the model proved to be proficient in selecting the most efficient placement in various geographical scenarios. However, in real-world applications, for example, at ITB Jatinangor, it must be acknowledged that the application will be more complicated because it requires measuring the distance between nodes with GIS and calculating activity levels accurately.

## ACKNOWLEDGMENT

The author deeply conveys heartfelt gratitude to Allah SWT, who has bestowed grace and blessings to complete this paper, titled "Optimal Placement of Waste Bin in ITB Jatinangor Using The  $p$ -Median Model in Graph Theory". The author would also extend appreciation to Dr. Rinaldi, as a lecturer of the IF1220 Discrete Mathematics course, for the dedication, guidance, encouragement, and motivation throughout the semester. Lastly, the author is genuinely thankful for the endless support and strength coming from the author's family and friends.

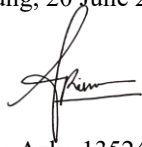
## REFERENCES

- [1] Kenneth H. Rosen, "Graph" in *Discrete Mathematics and Applications to Computer Science*, 8<sup>th</sup> Edition, New York: McGraw-Hill, pp. 673-676, 2019. [Online] Available: <https://www.chegg.com/textbooks/discrete-mathematics-and-its-applications-8th-edition-9781259731280-1259731286>. Accessed 15 June 2025 at 11:15 am.
- [2] Munir, Rinaldi, "Graf (Bag. 1)", 2024. [Online] Available: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf>. Accessed 15 June 2025 at 02:33 pm.
- [3] Daskin, M.S., *Network and Discrete Location*, 2<sup>nd</sup> Edition, Hoboken: John Wiley & Sons, 2013. [Online] Available: <https://www.wiley.com/en-us/Network+and+Discrete+Location%3A+Models%2C+Algorithms%2C+and+Applications%2C+2nd+Edition-p-9781118537015>. Accessed 16 June 2025 at 12:31 pm.
- [4] Marianov, V., & Serra, D., "Median Problems in Networks" in *International Series in Operations Research and Management Science*, pp. 39-59, 2006. [Online] Available: [https://doi.org/10.1007/978-1-4419-7572-0\\_3](https://doi.org/10.1007/978-1-4419-7572-0_3). Accessed 16 June 2025 at 01:46 pm.

# STATEMENT

I hereby declare that this paper is my original work, not an adaptation or translation of any other individual's work, and not plagiarism.

Bandung, 20 June 2025

A handwritten signature in black ink, appearing to read 'Arina', with a stylized flourish extending to the right.

Arina Azka 13524049