Solving the Set Covering Problem for Optimal Waste Bin Placement on Campus Using Branch and Bound Algorithm

Syaqina Octavia Rizha - 13524088
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524088@std.stei.itb.ac.id, [2]syaqinaoctavia@gmail.com

***Abstract*— Effective waste management infrastructure is essential for maintaining cleanliness and environmental sustainability in university campuses. This study addresses the problem of determining the optimal placement of waste bins across Institut Teknologi Bandung (ITB) Ganesha campus to ensure complete coverage of all buildings and activity zones while minimizing the number of bins required. The problem is formulated as a Set Cover Problem (SCP), a well-known NP-hard combinatorial optimization problem, where candidate bin locations must be selected such that every building or zone within the campus lies within a predefined coverage radius of at least one selected bin. To obtain an exact optimal solution, this research employs the Branch and Bound algorithm, which systematically explores the solution space while pruning branches that cannot yield better results than the current best solution, thereby avoiding the exponential time complexity of exhaustive search. Campus spatial data, including building coordinates and pedestrian distances within ITB Ganesha, were collected and modeled as a coverage graph to serve as the dataset for this study. The proposed algorithm was implemented and tested on this dataset to determine the minimum number of waste bins and their optimal locations. The results are compared with the current existing waste bin placement at ITB Ganesha to evaluate the efficiency and coverage improvement achieved by the proposed method. This study demonstrates that the Branch and Bound algorithm can produce an optimal and computationally feasible solution for facility placement problems in real-world campus environments..**

***Keywords*— Set Cover Problem, Branch and Bound Algorithm, Waste Bin Placement, Facility Location, Campus Optimization, NP-hard Problem.**

I. INTRODUCTION

Urban institutions such as university campuses face persistent challenges in maintaining environmental cleanliness and waste management efficiency as student populations and physical infrastructure continue to expand. At Institut Teknologi Bandung (ITB) Ganesha campus, the existing distribution of waste bins has largely developed through incremental and informal decision-making rather than systematic spatial planning. This often results in uneven coverage, where several buildings or activity zones are situated far from the nearest waste bin, while other areas exhibit redundant placement. Such inefficiency not only reduces the convenience and likelihood of proper waste disposal by students and staff, but also increases maintenance costs due to unnecessary bin redundancy. Addressing this issue requires a structured approach capable of determining the minimum number of waste bins, and their optimal locations, such that every building and zone within the campus is adequately covered.

This placement challenge can be formally represented as a Set Cover Problem (SCP), a well-established combinatorial optimization problem in computer science. In this formulation, the campus area is modeled as a set of target zones that must be covered, while each candidate bin location is represented as a subset capable of covering all zones within a predefined service radius. The objective is to select the minimum number of candidate locations such that the union of their coverage subsets encompasses the entire target set. Although conceptually straightforward, the Set Cover Problem is classified as NP-hard, meaning that an exhaustive search across all possible combinations becomes computationally infeasible as the number of candidate locations and target zones increases. This computational barrier necessitates an algorithmic strategy capable of finding the exact optimal solution without exploring the entire solution space exhaustively.

To address this challenge, this paper proposes the application of the Branch and Bound algorithm to determine the optimal placement of waste bins across ITB Ganesha campus. Unlike approximation-based heuristics, Branch and Bound systematically explores the solution space while employing a bounding function to prune branches that cannot yield a better solution than the current best-known result, thereby guaranteeing optimality while significantly reducing computational overhead compared to brute-force enumeration. The spatial data of ITB Ganesha, including building locations and pedestrian distances, is modeled as a coverage graph to serve as the dataset for this study. By applying the proposed algorithm to this real-world campus environment, this paper aims to provide a computationally efficient and exact solution for facility placement problems, offering a practical framework for improving waste management infrastructure planning in university settings.

II. THEORETICAL FOUNDATIONS

A. Set Covering Problem

The set covering problem is a significant NP-hard problem in combinatorial optimization. Given a collection of elements, the set covering problem aims to find the minimum number of sets that incorporate (cover) all of these elements.

Problem formulation: In the set covering problem, two sets are given: a set U of elements and a set S *o*f subsets of the set U. Each subset in S is associated with a predetermined cost, and the union of all the subsets covers the set U. This combinatorial problem then concerns finding the optimal number of subsets whose union covers the universal set while minimizing the total cost.

The Set Covering Problem (SCP) is defined on a universe of elements $U = \{u_1, u_2, \dots, u_m\}$ and a collection of subsets $S = \{s_1, s_2, \dots, s_n\}$. Each subset $s_i \subseteq U$ is associated with a positive cost $c_i > 0$. The union of all subsets in S covers the entire universe, i.e., $\bigcup_{i=1}^{n} s_i = U$ and each subset contains at least one element of U. The objective of the SCP is to determine a minimum-cost collection of subsets $X \subseteq S$ such that every element in the universe U is covered by at least one subset in X.

- Integer linear program formulation:

  An integer linear program (ILP) model can be formulated for the minimum set covering problem as follows:

  Decision variables:

$$y_i = \begin{cases} 1, & \text{if subset } i \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

  Objective function:

$$\min \sum_{i=1}^{n} c_i \, y_i$$

  Constraints:

$$\sum_{i=1}^{n} a_{ij} \, y_i \geq 1, \quad \forall j = 1,2,\dots,m$$

$$y_i \in \{0,1\}, \quad \forall i = 1,2,\dots,n$$

  The objective function $\sum_{i=1}^{n} c_i \, y_i$ is defined to minimize the number of subset s_i that cover all elements in the universe by minimizing their total cost. The first constraint implies that every element i in the universe U must be be covered and the second constraint $y_i \in \{0,1\}$ indicates that the decision variables are binary which means that every set is either in the set cover or not.

Set covering problems are significant NP-hard optimization problems, which implies that as the size of the problem increases, the computational time to solve it increases exponentially. Therefore, there exist approximation algorithms that can solve large scale problems in polynomial time with optimal or near-optimal solutions.

B. Branch and Bound Algorithm

Branch and bound algorithms are used to find the optimal solution for combinatory, discrete, and general mathematical optimization problems.

- Branching: The main problem is broken into smaller subproblems, forming a tree-like structure where each node represents a partial solution.
- Bounding: For each node, the algorithm computes an upper or lower bound on the best possible solution within that subproblem. If this bound is worse than the current best-known solution, the node is pruned (i.e., not explored further).
- Pruning: By eliminating branches that cannot yield better solutions than the current best, the algorithm reduces the number of computations, enhancing efficiency.

This process continues until the best solution is found or all branches have been explored.

Basic Concepts:

- **Generation of a state space tree**: As in the case of backtracking, B & B generates a state space tree to efficiently search the solution space of a given problem instance. In B & B, all children of an E-node in a state space tree are produced before any live node gets converted in an E-node. Thus, the E-node remains an E-node until i becomes a dead node.
- **Evaluation of a candidate solution**: Unlike backtracking, B & B needs additional factors evaluate a candidate solution:
  1. A way to assign a bound on the best values of the given criterion functions to each node in a state space tree: It is produced by the addition of further components to the partial solution given by that node.
  2. The best values of a given criterion function obtained so far: It describes the upper bound for the maximization problem and the lower bound for the minimization problem.

It optimizes the search for a solution vector in the solution space of a given problem instance.

While Backtracking systematically explores the solution space by pruning branches that violate problem constraints, Branch and Bound extends this approach by incorporating a bounding function that estimates the best possible outcome achievable from a given branch. This allows the algorithm to prune branches that, while still valid, cannot yield a solution better than the current best-known result, thereby guaranteeing global optimality rather than merely finding a feasible solution.

C. Graph Theory

1. Bipartite Graph

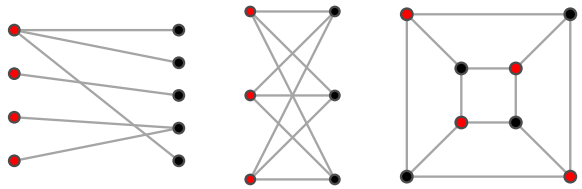

Fig.1. Bipartite Graph example source (https://mathworld.wolfram.com/BipartiteGraph.html)

A bipartite graph, also called a bigraph, is a set of graph vertices decomposed into two disjoint sets such that no two graph vertices within the same set are adjacent. The two vertex classes are said to form a bipartition. A bipartite graph is a special case of a k-partite graph with $k = 2$. The illustration above shows some bipartite graphs, with vertices in each graph colored based on to which part of the bipartition they belong. Bipartite graphs are equivalent to two-colorable graphs. All acyclic graphs are bipartite. A cyclic graph is bipartite iff all its cycles are of even length.

2. Matching and Vertex Cover

- Matching: a set of edges without common vertices. In other words, a subset of the edges is a matching if each vertex appears in at most one edge of that matching.
- Vertex cover: subset of the nodes that together touch all the edges

D. Facility Location Problem

Facility Location Problem (FLP) is a class of optimization problems concerned with determining the most appropriate locations for facilities in order to serve a set of demand points efficiently. The primary objective of FLP is to identify facility locations that optimize a particular performance measure, such as minimizing transportation costs, minimizing service distance, maximizing service coverage, or balancing resource allocation. Facility location models have been widely applied in various domains, including healthcare facilities, logistics centers, warehouses, emergency service stations, public transportation hubs, and waste management systems.

In facility location analysis, the problem typically consists of three main components: facilities, demand points, and a performance criterion. Facilities represent service providers, while demand points represent locations that require service. The relationship between facilities and demand points is often modeled using distance, travel time, service cost, or coverage measures. The goal is to determine facility locations that provide the desired level of service while satisfying operational constraints.

Set Covering Problem (SCP) is one of the most important variants of the Facility Location Problem. Unlike traditional facility location models that focus on minimizing travel distance or transportation cost, SCP aims to identify the minimum number of facilities required to cover all demand points within a predefined service radius. In this model, a demand point is considered covered if at least one facility is located within the specified coverage distance. Therefore, SCP is particularly suitable for applications where complete service coverage is more important than minimizing average travel distance.

III. IMPLEMENTATION

A. System Architecture and Components

The waste bin placement optimization program is implemented in Python, chosen for its extensive ecosystem of scientific computing and geospatial libraries. The pipeline is divided into two sequential scripts:

- data_collection.py: Handles data acquisition and coverage modeling.
- branch_and_bound.py: Executes the optimization algorithm and generates result visualizations.

Libraries Used

- OSMnx: Retrieves pedestrian road network data from OpenStreetMap, specifically for the ITB Ganesha campus.
- NetworkX: Serves as the underlying graph structure (representing intersections as nodes and road segments as weighted edges).
- NumPy: Constructs and manipulates the binary coverage matrix and supports bitmask-based computations.
- Matplotlib: Serves as the primary visualization library for generating static maps, matrix heatmaps, and state-space tree diagrams.

B. Data Collection & Coverage Modeling

The process begins in data_collection.py through the following steps:

1. Road Network Extraction: The pedestrian network is fetched using `ox.graph_from_place()` with `network_type = 'walk'`. The graph is then projected into a metric coordinate reference system (UTM) using `ox.project_graph()` to ensure accurate distance measurements in meters.
2. Coordinate Definition:
   - Target Zones: Key campus areas (lecture buildings, the central library, canteens, laboratories, and parking lots).
   - Candidate Locations: Strategically chosen points for potential bin placement (building entrances, pedestrian intersections, and seating areas).
   - Note: Coordinates are entered manually as GPS latitude-longitude pairs obtained directly from Google Maps.
3. Binary Coverage Matrix ($n_{\text{candidates}} \times n_{\text{zones}}$):
   - Pairwise distances between every candidate and target zone are computed using the Haversine

formula.

- If the distance falls within the defined coverage radius (40 meters), the cell is assigned a value of 1 (covered); otherwise, it is set to 0.
- The matrix and coordinate data are saved to campus_data.json for the next stage.

C. *Branch and Bound Optimization*

The second script, branch_and_bound.py, loads the JSON data and applies the Branch and Bound algorithm to solve the Set Covering Problem.

Acceleration & Search Strategy

- Bitmask Optimization: Each candidate's coverage row is pre-converted into an integer bitmask. Coverage union operations are performed using bitwise OR, which is significantly faster than array-based set operations.
- Best-First Search: Tree traversal is managed via a min-heap priority queue. Each node in the state-space tree represents a binary decision: whether a candidate location is "Included" or "Excluded" from the solution.
- Lower Bound Function:

$$\text{Lower Bound} = \text{Selected Bins} + \left\lceil \frac{\text{Remaining Uncovered Zones}}{\text{Max. New Coverage by Remaining Candidate}} \right\rceil$$

- Pruning: If a node's lower bound is greater than or equal to the cost of the best solution found so far, the node is pruned, and its entire subtree is discarded.
- Validation Method: For comparison, if the number of candidates is ≤ 20, a Brute-Force Exhaustive Search is run using combinations to guarantee and verify global optimality.

D. *Visualization and Results Reporting*

The final output is presented through three primary visualization panels and an auxiliary tree diagram.

- Campus Map: Renders the selected bin locations as filled triangular markers with circular coverage areas overlaid, alongside target zones and unselected candidates.
- Reduced Coverage Matrix: Displays a matrix containing only the selected candidates, annotated with checkmarks to confirm that every target zone column receives at least one coverage entry.
- State-Space Tree Diagram: A separate visualization rendering the first four levels of the decision tree. Nodes are color-coded to clearly distinguish between active exploration nodes, feasible solutions, and pruned nodes.

IV. TESTING AND RESULT

A. *Geospatial Network Initialization and Coordinate Mapping*

The initial phase of the implementation establishes the spatial environment by fetching the pedestrian road network of the ITB Ganesha campus via OpenStreetMap. As displayed in Fig.2, the system constructs a base topological layout consisting of walkable paths, ignoring restricted or non-pedestrian roads.Superimposed onto this projected network are two manually defined point sets: the green squares represent the target zones (key university buildings and high-traffic student hubs such as TVST, Oktagon, Labtek, and Fidas), while the orange triangles signify the 38 distinct candidate locations $C01\ to\ C38$ proposed for potential waste bin installation. This baseline configuration visualizes the raw geospatial distribution of demand versus supply constraints before any mathematical optimization is enforced.

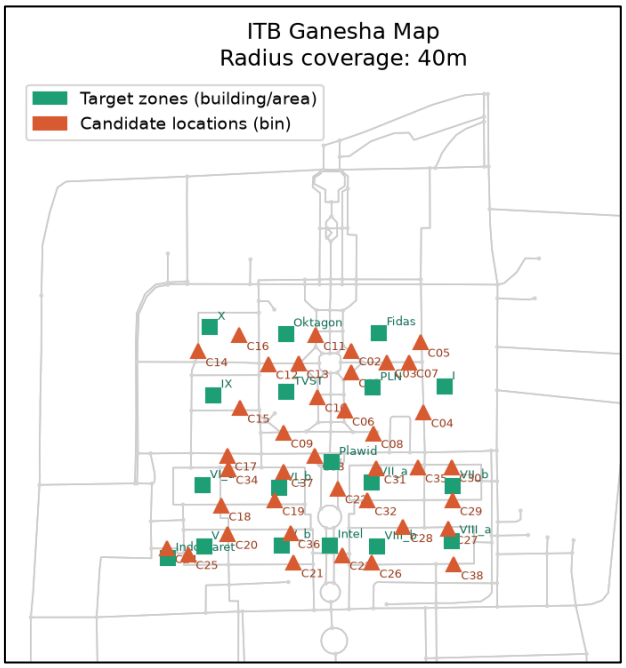

Fig.2. ITB Ganesha map with selected building and bin installation candidate points.
source (Author)

B. *Initial Coverage Matrix Construction*

With the geographic coordinates mapped, the system computes the great-circle distances between all pairs of candidate locations and target zones utilizing the Haversine formula. Applying a strict maximum coverage radius threshold of 40 meters, the continuous distance metrics are converted into a discrete logical structure.The resulting initial coverage matrix is illustrated as a comprehensive heatmap in Fig.3. In this visualization, a dark green block populated with a value of 1 indicates that a specific candidate location successfully covers a given target zone within the allowable 40-meter walking range, whereas a light green block with a value of 0 denotes a lack of spatial coverage. This $38 \times n_{\text{ZONES}}$ binary matrix serves as the comprehensive algebraic input required by the combinatorial solver.

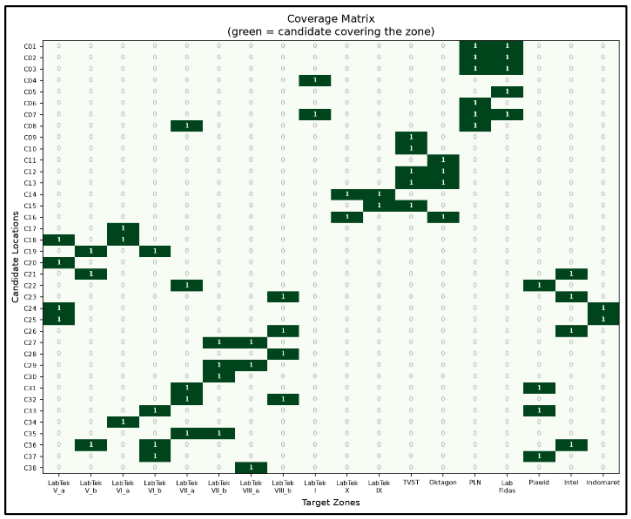

Fig.3. Initial Coverage Matrix.
source (Author)

C. *Optimal Waste Bin Allocation and Spatial Coverage Visualization*

By processing the binary coverage matrix through the Best-First Search Branch and Bound framework, the program isolates the globally optimal subset of waste bin locations. Out of the 38 original candidates, the algorithm successfully minimizes the facility count down to exactly 9 optimal location C07,C15,C16,C24,C29,C32,C33,C34 and C36 .As shown in Fig.4, the selected bins are rendered as distinct orange triangles enclosed by translucent pink circles representing their 40-meter radius coverage boundaries. Unselected candidates are faded out as light gray triangles. The spatial plot provides visual proof of the algorithm's mathematical efficacy, demonstrating that the overlapping 40-meter boundaries seamlessly cover every critical target zone across the campus layout with zero dead zones.

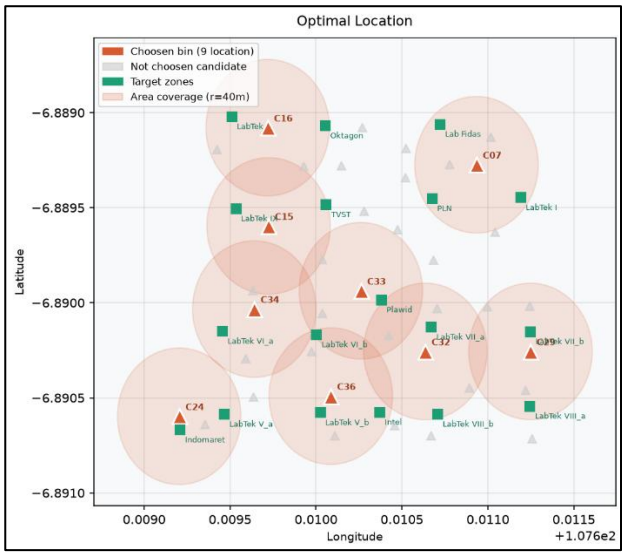

Fig.4. Geographic distribution and 40-meter radius coverage areas.
source (Author)

D. *Reduced Optimal Coverage Matrix Verification*

To mathematically validate that the 9 chosen facilities satisfy the entire demand space, the original data structure is filtered down into a reduced coverage matrix. As shown in Fig.5, this clean matrix isolate contains only the rows corresponding to the 9 selected optimal bins.Each column represents a target zone, and the matrix is explicitly annotated with green checkmarks (V) at the intersecting cells where coverage is established. The visual output confirms that every single column receives at least one checkmark, proving that the Set Covering Problem's primary constraint, ensuring 100% total campus coverage, is fully met without redundant asset allocation.

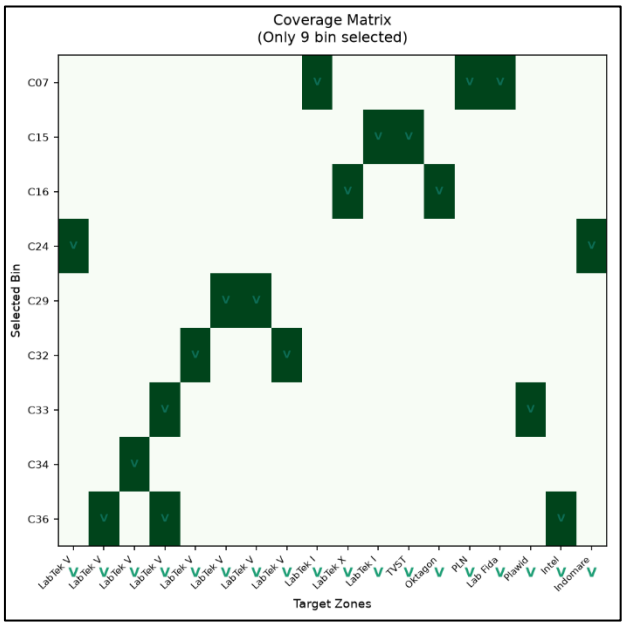

Fig.5. Filtered coverage matrix.
source (Author)

E. *State-Space Tree Exploration and Pruning Dynamics*

The computational mechanics and pruning efficiency of the Branch and Bound algorithm are captured through a hierarchical state-space tree diagram. As illustrated in Fig.6, the visualization tracks the first 4 decision levels of the tree structure branching out from the initial root node (Level 0) down through the first 38 candidate choices.

Each node contains an integer that details the cumulative number of bins selected along that specific decision path. Moving down a left branch denotes the inclusion of a candidate (Y), which increments the bin counter, whereas moving down a right branch denotes an exclusion (X), keeping the bin count constant. The balanced nature of the tree in these early stages reflects the active exploration phase before the lower bound heuristic encounters a high-quality upper bound solution, after which massive sub-tree discarding (pruning) occurs in the deeper segments of the execution.

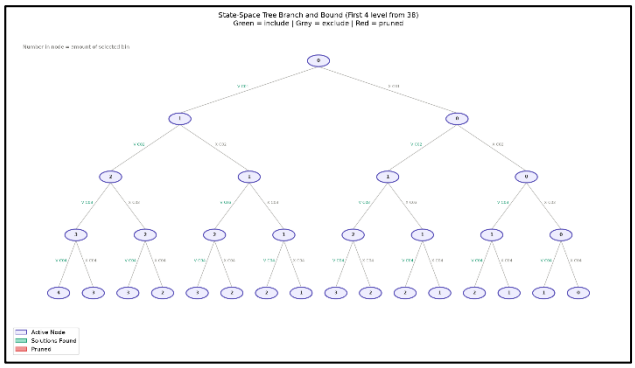

Fig.6. State-space decision tree diagram for the first 4 levels. source (Author)

V. CONCLUSION

This study demonstrates that formulating campus waste bin placement as a Set Covering Problem (SCP) provides a mathematically rigorous foundation for optimizing facility allocation. By utilizing the Best-First Search Branch and Bound algorithm on the ITB Ganesha pedestrian network, the system successfully bypassed the computational limits of exhaustive search through look-ahead lower bound pruning. The algorithm achieved a globally optimal solution, minimizing the required infrastructure from 38 candidate locations down to exactly 9 strategic positions while maintaining 100% total coverage across all critical university target zones.

The integration of bitmask optimization with a min-heap priority queue proved essential for accelerating state-space tree traversals and ensuring computational efficiency. Practically, this automated, data-driven approach allows campus logistics teams to replace arbitrary or reactive deployment strategies with optimal, systematic placement blueprints. The resulting blueprint maximizes the efficiency of campus cleanliness infrastructure by utilizing overlapping service boundaries to eliminate dead zones and prevent costly asset redundancy.

Despite its success, this framework is limited by its static nature, relying on a uniform 40-meter radius and fixed geographic coordinates that do not reflect real-world operational changes. It lacks accommodation for dynamic factors such as fluctuating student densities, daily waste accumulation rates, and varying path topography. Future work will focus on expanding this model to incorporate dynamic capacity constraints, real-time IoT fill-level sensors, and weighted demand parameters to better capture the complexities of campus waste logistics.

VI. APPENDIX

The full codebase for this implementation can be found on GitHub.

VII. ACKNOWLEDGMENT

The author wishes to express the deepest gratitude to the Lord Almighty for His guidance and blessings throughout the research and writing of “Solving the Set Covering Problem for Optimal Waste Bin Placement on Campus Using Branch and Bound Algorithm”. Through His grace, perseverance was granted to navigate the complexities of this project and bring it to a successful conclusion. Heartfelt gratitude is also extended to all who have supported this endeavor:

1. Ir. Rila Mandala, M.Eng., Ph.D., as the lecturer for the K2 IF2211 Algorithms Strategies course, for his essential guidance in applying abstract graph theory and implementation of algorithm startegies in many cases.
2. The author’s parents, for their endless support, prayers, and motivation.
3. Friends and colleagues, for the constructive discussions and encouragement that were vital during the development of this paper. The completion of this work would not have been possible without their invaluable contributions.

The completion of this work would not have been possible without their invaluable contributions.

VIII. REFERENCES

[1] Cornell University Optimization Wiki, “Set covering problem,” [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Set_covering_problem. [Accessed: 18-Jun-2026].

[2] GeeksforGeeks, “Introduction to branch and bound - data structures and algorithms tutorial,” [Online]. Available: https://www.geeksforgeeks.org/dsa/introduction-to-branch-and-bound-data-structures-and-algorithms-tutorial/. [Accessed: 18-Jun-2026].

[3] E.W. Weisstein, “Bipartite graph,” From MathWorld--A Wolfram Web Resource. [Online]. Available: https://mathworld.wolfram.com/BipartiteGraph.html. [Accessed: 18-Jun-2026].

[4] A.A. Ahmadi, “ORF523: Lecture 6,” Princeton University Course Documentation. [Online]. Available: https://www.princeton.edu/~aaa/Public/Teaching/ORF523/ORF523_Lec6.pdf. [Accessed: 19-Jun-2026].

[5] Cornell University Optimization Wiki, “Facility location problem,” [Online]. Available: https://optimization.cbe.cornell.edu/index.php?title=Facility_location_problem. [Accessed: 19-Jun-2026].

[6] Defense Technical Information Center (DTIC), “Solving set covering problem using heuristics with branch and bound” [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA151905.pdf. [Accessed: 19-Jun-2026].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Syaqina Octavia Rizha