Branch-and-Bound for Van-Drone Collaborative Routing: Experiments on Real-World VDRPMDPC Instances

Mahesa Fadhillah Andre - 13523140
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: mahesa0208@gmail.com , 13523140@std.stei.itb.ac.id

***Abstract*—This paper proposes a solution for the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC) using the Branch-and-Bound algorithm. The aim is to find an optimal delivery plan that combines van and drone operations while minimizing the total delivery time. The problem considers constraints such as drone delivery capacity and maximum flight distance. A greedy van-only approach is used as the initial upper bound to improve the efficiency of pruning in the Branch-and-Bound process. The algorithm is tested on real-world inspired distance matrices with configurable parameters, including van speed, drone speed, and service time. The results show that the proposed method can find optimal delivery plans while exploring significantly fewer nodes compared to exhaustive search methods. These findings demonstrate the potential of Branch-and-Bound for solving VDRPMDPC in small to medium-scale delivery scenarios.**

Keywords—Branch-and-Bound; van-drone routing; VDRPMDPC; optimization; delivery planning.

I. INTRODUCTION

The increasing demand for rapid and reliable delivery services, driven by the rise of e-commerce and modern retail, has placed significant pressure on last-mile delivery operations. Traditional delivery methods that rely solely on vans or trucks are often unable to meet these demands efficiently due to various challenges. Traffic congestion, limited delivery time windows, high fuel consumption, and operational costs are some of the key issues faced in urban and suburban delivery networks. To address these limitations, hybrid delivery systems that combine ground vehicles, such as vans, with unmanned aerial vehicles (drones) have been proposed and studied. In such systems, the van serves as a mobile depot that carries drones to strategic locations. From these locations, drones are launched to deliver packages directly to customers, bypassing obstacles on the road network and reducing delivery time.

The adoption of hybrid van-drone delivery systems presents both opportunities and challenges. On the one hand, this collaboration can significantly improve delivery efficiency by leveraging the respective strengths of vans and drones. On the other hand, planning the optimal delivery routes and schedules for both the van and its drones introduces considerable complexity. The delivery plan must determine not only the sequence of van stops but also when and where drones should be deployed, as well as which delivery points should be assigned to each drone. The problem becomes more difficult when operational constraints are introduced. These include limitations on drone delivery capacity, maximum drone flight distance, service times required at each delivery point, and the need to synchronize the van's position with drone operations for launch and retrieval.

This problem is referred to as the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC). The objective is to minimize the total delivery time while satisfying all operational constraints related to both van and drone deliveries. The VDRPMDPC is classified as an NP-hard problem because the number of possible delivery sequences increases exponentially with the number of delivery points. As a result, exact solutions become computationally expensive for larger instances, and careful algorithmic design is required to solve the problem efficiently.

Various solution techniques have been applied to address similar routing problems. Heuristic methods, such as genetic algorithms, simulated annealing, and ant colony optimization, can generate near-optimal solutions within reasonable computation times, making them suitable for large-scale instances. However, these methods do not provide guarantees of solution optimality. In contrast, exact algorithms, including dynamic programming and Branch-and-Bound, are designed to find the optimal solution by systematically exploring the solution space. Among these, Branch-and-Bound is widely used because it allows pruning of non-promising branches based on bounds, thereby reducing the number of solutions that must be fully explored. This technique ensures that the best solution is found while avoiding unnecessary computations.

In this study, a Branch-and-Bound algorithm is applied to solve the VDRPMDPC. To enhance the efficiency of the search process, a greedy van-only baseline is computed first to serve as an initial upper bound. This baseline is obtained by having the van visit the nearest unvisited client at each step without deploying drones. The Branch-and-Bound algorithm then explores alternative solutions involving van moves and

drone deployments. At each stage, branches that cannot yield a better solution than the current best-known delivery time are pruned to save computation time.

The implementation of the algorithm includes configurable parameters such as van speed, drone speed, service time, drone delivery capacity, maximum drone distance, and the number of delivery points. The distance data used in the experiments are derived from a real-world inspired distance matrix to ensure that the scenarios reflect practical conditions. The results of the study show that the Branch-and-Bound algorithm is capable of finding optimal delivery plans efficiently for small to medium-sized problem instances. The use of the greedy baseline significantly reduces the number of nodes explored compared to a brute-force search, highlighting the effectiveness of the proposed approach for solving the VDRPMDPC.

II. Theoretical Basis

A. *Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC)*

The Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC) is a complex variant of the classic Vehicle Routing Problem (VRP). In this problem, a van and a set of drones are used together to perform last-mile deliveries to multiple clients. The van acts as both a carrier for packages and as a mobile base station from which drones are launched and retrieved. The drones assist the van by delivering packages directly to clients within their operational range, allowing faster and more flexible delivery compared to van-only routes.

Each client location must be served exactly once, either by the van or by one of the drones. The van can move freely between client locations using the road network, while each drone is constrained by several factors. First, the drone's delivery capacity limits the number of packages it can carry in a single mission. Second, the drone's maximum flight distance defines how far it can travel from the van's current location and still be able to return safely. Third, each delivery action requires a specific service time, both for van deliveries and for the time needed to launch and retrieve the drones.

The objective of the VDRPMDPC is to determine the sequence of van moves and drone dispatches that minimizes the total delivery time. This includes travel time, service time at each client, and any additional time associated with drone operations, such as launch and retrieval. The problem is combinatorially complex because each client can potentially be served by either the van or a drone, and the number of possible sequences increases factorially with the number of delivery points. As a result, the VDRPMDPC belongs to the class of NP-hard problems, and solving it exactly requires sophisticated search techniques.

This type of routing problem is highly relevant to modern logistics, where van-drone collaboration is seen as a promising strategy to improve delivery efficiency in urban and suburban areas. Solutions to the VDRPMDPC can help reduce delivery times, lower operational costs, and improve customer satisfaction in real-world delivery networks.

B. *Branch-and-Bound Algorithm*

The Branch-and-Bound (BnB) algorithm is an exact method that is widely used to solve combinatorial optimization problems where the solution space is large and complex. The key concept of BnB is to represent the solution space as a state-space tree. In this tree, each node corresponds to a partial solution that can be extended by exploring its branches. The root node represents the initial state where no delivery decisions have been made. Each level in the tree represents the addition of one more decision, such as choosing the next delivery point or dispatching a drone batch.

As the algorithm progresses, the solution space is explored systematically by branching into possible options at each node. However, not all branches are worth exploring. To avoid unnecessary computation, a bound is calculated at each node. The bound represents the minimum possible cost (such as delivery time) that could be achieved if that partial plan were expanded into a complete plan. If the bound is greater than or equal to the cost of the best complete solution found so far, the node is pruned because it cannot lead to an improvement. This process ensures that only promising branches are explored, and the search remains efficient despite the complexity of the solution space.

The general structure of a state-space tree used in Branch-and-Bound is illustrated below. In this tree, branching represents decisions, and pruning is performed at nodes where the bound exceeds the best known cost.

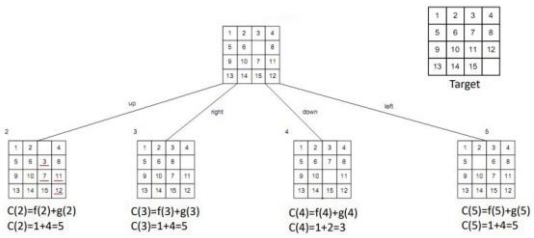


Fig. 2.1. Example of a state-space tree illustrating branching decisions and pruning in Branch-and-Bound (Source: Branch-and-Bound Algorithm Part 1, Slide by Dr. Rinaldi Munir).

In the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC), the application of Branch-and-Bound involves representing each node in the state-space tree as a partial delivery plan. This plan includes the sequence of van stops and drone batch dispatches that have been executed so far. The cost associated with a node is the total delivery time accumulated up to that point, which includes van travel time, drone flight time, launch and retrieval time, and service time at delivery points. The branching at each node considers two types of decisions: either the van proceeds to an undelivered client, or a batch of drones is dispatched to deliver packages to clients within the drone's capacity and distance limits.

The diagram below provides an example of how branching decisions are represented when applying the Branch-and-Bound algorithm to van-drone collaborative delivery.

At each branching point, the cumulative delivery time of

the partial plan is updated, and a bound is calculated by adding an optimistic estimate of the minimum remaining delivery time needed to complete the plan. This bound is compared to the best delivery time found so far. If the bound exceeds the best known time, the branch is pruned, and no further exploration is conducted from that node. This mechanism significantly reduces the number of nodes explored compared to a brute-force exhaustive search.

The performance of the Branch-and-Bound algorithm is highly dependent on the quality of the bound used. A tight bound allows the algorithm to prune more nodes, thereby reducing the computational load. In this study, a greedy van-only delivery plan is computed at the start of the algorithm to provide an initial upper bound. This plan consists of the van visiting the nearest unvisited client at each step, without deploying drones. Although this plan does not yield the optimal solution, it offers a quick estimate that enables more aggressive pruning during the Branch-and-Bound search process.

The combined use of the state-space tree, bounding, and pruning ensures that the algorithm finds the optimal solution while maintaining computational efficiency. This approach is particularly effective for small to medium-sized VDRPMDPC instances where exact solutions are preferred over heuristic approximations.

C. *Greedy Baseline as Initial Upperbound*

The greedy baseline is used in this study as an initial upper bound to enhance the performance of the Branch-and-Bound algorithm. This baseline is constructed by generating a simple van-only delivery plan in which the van visits the nearest unvisited client at each step until all clients are served. The plan is determined quickly by applying the nearest-neighbor heuristic, which is commonly used in the Travelling Salesman Problem (TSP) and related routing problems. The total delivery time of this greedy plan includes both travel time and service time at each client.

Although the greedy baseline does not provide an optimal solution, it offers a feasible delivery plan that can be computed with minimal effort. The total time required by this plan serves as the initial upper bound in the Branch-and-Bound search. As the search progresses, any partial plan whose cumulative cost exceeds this upper bound can be pruned immediately, since it cannot lead to a better solution. This strategy significantly reduces the number of nodes that need to be explored in the state-space tree.

An example of a greedy van-only route can be visualized as a simple sequence where the van proceeds from the depot to the nearest client, then to the next nearest client, and so on, until all clients have been served.

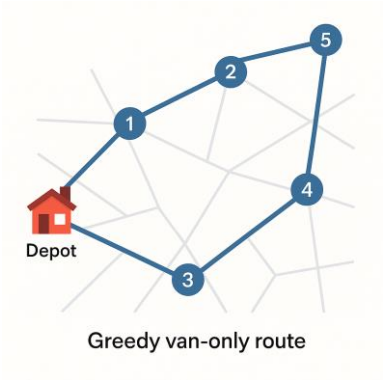


Fig. 2.2. Example of a greedy van-only delivery route on a simplified delivery map, illustrating the nearest-neighbor approach (custom illustration for this study).

The use of the greedy baseline is especially valuable in small to medium-sized instances of the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC), as it provides an effective reference for pruning without the need for complex preliminary computation.

D. *Application of Branch-and-Bound to VDRPMDPC*

In the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC), the Branch-and-Bound algorithm is applied to explore possible delivery plans that combine van movements and drone dispatches. Each node in the state-space tree represents a partial delivery plan that includes the van's current position, the sequence of van deliveries made so far, and the sets of clients served by drones. The goal at each node is to extend the partial plan by either moving the van to the next client or launching a batch of drones to deliver to nearby clients.

The branching process at each node follows two possible actions. The first action involves the van proceeding to an undelivered client and performing the delivery, which adds travel time and service time to the cumulative delivery cost. The second action involves dispatching a batch of drones to serve clients that are within the drone's maximum distance from the van's current location and within the drone's delivery capacity. The cost of this action includes the time needed to launch the drones, the maximum flight time required by the batch, and the retrieval time.

At each stage of the search, the cumulative delivery time of the partial plan is computed. A bound is then determined by adding an optimistic estimate of the minimum remaining time needed to complete the plan.

The bound at a node can be expressed as:

$$\text{Bound} = \text{CumulativeTime} + \text{EstimatedRemainingTime}$$

where CumulativeTime is the total time of the partial plan up to the current node, and EstimatedRemainingTime represents an optimistic estimate of the minimum time required to complete the remaining deliveries.

If the sum of the cumulative time and the bound exceeds the best complete delivery time found so far, the node is

pruned. This pruning mechanism ensures that the search focuses only on promising branches and avoids unnecessary computations.

The figure below illustrates how van and drone actions expand the state-space tree in this problem.

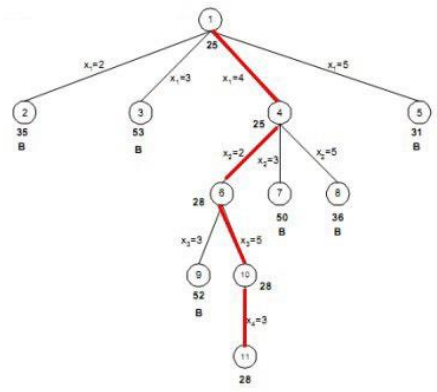


Fig. 2.3. Example of van-drone branching decisions in the state-space tree (Source: Branch-and-Bound Algorithm Part 2, slide by Dr. Rinaldi Munir).

The efficiency of this approach depends not only on the branching and pruning strategies but also on the quality of the initial upper bound provided by the greedy baseline. The tighter the initial bound, the more branches can be pruned early in the search. This combined strategy allows the Branch-and-Bound algorithm to produce optimal delivery plans for VDRPMDPC instances involving small to medium numbers of clients, while keeping the computational cost reasonable.

III. Implementation Method

The proposed solution for the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC) was implemented using the Python programming language. The algorithm was designed to allow flexibility through configurable parameters entered by the user at runtime. These parameters include van speed, drone speed, service time at each delivery point, drone delivery capacity, the number of drones available, and the maximum distance that drones can travel on a single flight. The number of client locations used in each test scenario is also configurable to simulate instances of different sizes.

A. Data Input and Preprocessing

Distance data between the depot and client locations were stored in an Excel file in matrix form. This matrix represents the real-world distances between locations in a simulated urban environment. The program reads this matrix using the pandas library, ensuring that only valid numeric data are processed by dropping empty rows and columns. The data are trimmed to include only the depot and the specified number of client locations as configured by the user.

Two time matrices are then constructed: one for the van and one for the drone. The time required to travel between any two locations is computed by dividing the distance by the respective vehicle's speed and converting the result into minutes. This ensures that the travel times used in the algorithm reflect the configured vehicle speeds accurately.

```python
import pandas as pd
import os

# — Configuration (user inputs) ————————————————————————
VAN_SPEED_KMH    = int(input("Enter van speed (km/h, default 30): ") or 30)
DRONE_SPEED_KMH = int(input("Enter drone speed (km/h, default 50): ") or 50)
SERVICE_TIME     = float(input("Enter service time (minutes, default 0.5): ") or 0.5)
DRONE_CAPACITY  = int(input("Enter drone capacity (deliveries per drone, default 3): ") or 3)
NUM_DRONES       = int(input("Enter number of drones available (default 3): ") or 3)
DRONE_MAX_DIST  = float(input("Enter drone max distance (km, default 3.0): ") or 3.0)
MAX_HOUSES       = int(input("Enter max houses to read (default 8): ") or 8)

# — Utility: load only first MAX_HOUSES sites ——————————————
def load_data(filename):
    base = os.path.dirname(os.path.abspath(__file__))
    data_dir = os.path.join(base, '..', 'data')
    df = pd.read_excel(os.path.join(data_dir, filename), index_col=0)
    df = df.apply(pd.to_numeric, errors='coerce')
    df = df.dropna(how='all', axis=0).dropna(how='all', axis=1)
    labels = [str(i) for i in df.index if str(i) in df.columns]
    labels = labels[:MAX_HOUSES]
    mat = df.loc[labels, labels].values
    print(f"Loaded locations: {labels}")
    return labels, mat

# — Build time matrices (km → minutes) ————————————————————
def make_time_matrix(dist, speed):
    tm = dist / speed * 60
    print(f"Computed time matrix at {speed} km/h")
    return tm
```

Fig. 3.1. Code snippet showing data input and preprocessing logic (Source: program developed for this study).

B. Greedy Baseline Computation

A greedy van-only delivery plan is computed as an initial upper bound for the Branch-and-Bound algorithm. In this plan, the van always proceeds to the nearest unvisited client at each step until all deliveries are completed. The total delivery time of this greedy plan includes both travel time and service time at each client. Although this method does not produce an optimal solution, it provides a fast estimate of a feasible plan that enables more effective pruning during the Branch-and-Bound search.

```python
# — Greedy van-only as baseline without lambda ————————————
def greedy_van(locations, van_time):
    # Helper to compute cost from current position
    def cost_to(current_pos, j):
        return van_time[current_pos, j]

    visited = set()
    pos = 0  # depot index
    order = []
    total = 0.0
    n = len(locations)
    while len(visited) < n - 1:
        # collect unvisited choices
        choices = [i for i in range(1, n) if i not in visited]
        # pick best_choice manually
        best_choice = None
        best_cost = float('inf')
        for j in choices:
            c = cost_to(pos, j)
            if c < best_cost:
                best_cost = c
                best_choice = j
        next_i = best_choice
        # update time and state
        total += van_time[pos, next_i] + SERVICE_TIME
        order.append(locations[next_i])
        visited.add(next_i)
        pos = next_i
    print(f"Greedy route: {'->'.join(order)} = {total:.2f} min")
    return total
```

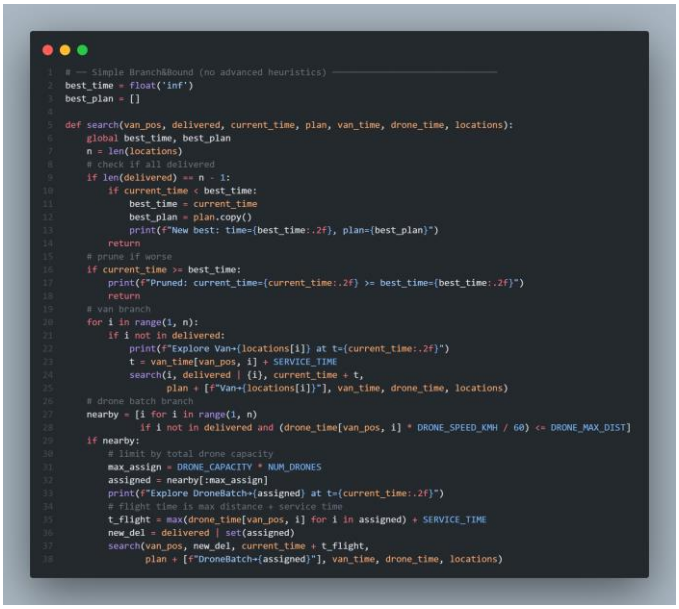
Fig. 3.2. Code snippet showing greedy van-only baseline computation (Source: program developed for this study).

C. Branch-and-Bound Search Process

The Branch-and-Bound search explores the solution space by recursively generating partial delivery plans. Each node in the search tree represents a state that includes the van's current location, the set of clients that have been served, and the sequence of actions taken so far. At each node, two types of branching are performed:

- The van proceeds to an undelivered client, adding travel and service time to the cumulative cost.
- A batch of drones is dispatched to serve clients within the maximum distance and capacity limits, adding launch, flight, and retrieval time to the cumulative cost.

After each branching decision, the cumulative delivery time is updated. A bound is computed by adding an optimistic estimate of the minimum time required to complete the plan. If the sum of the cumulative time and the bound exceeds the best complete delivery time found so far, the branch is pruned.



```python
# — Simple Branch&Bound (no advanced heuristics) ————————————————
best_time = float('inf')
best_plan = []

def search(van_pos, delivered, current_time, plan, van_time, drone_time, locations):
    global best_time, best_plan
    n = len(locations)
    # check if all delivered
    if len(delivered) == n - 1:
        if current_time < best_time:
            best_time = current_time
            best_plan = plan.copy()
            print(f"New best: time={best_time:.2f}, plan={best_plan}")
        return
    # prune if worse
    if current_time >= best_time:
        print(f"Pruned: current_time={current_time:.2f} >= best_time={best_time:.2f}")
        return
    # van branch
    for i in range(1, n):
        if i not in delivered:
            print(f"Explore Van→{locations[i]} at t={current_time:.2f}")
            t = van_time[van_pos, i] + SERVICE_TIME
            search(i, delivered | {i}, current_time + t,
                   plan + [f"Van→{locations[i]}"], van_time, drone_time, locations)
    # drone batch branch
    nearby = [i for i in range(1, n)
              if i not in delivered and (drone_time[van_pos, i] * DRONE_SPEED_KMH / 60) <= DRONE_MAX_DIST]
    if nearby:
        # limit by total drone capacity
        max_assign = DRONE_CAPACITY * NUM_DRONES
        assigned = nearby[:max_assign]
        print(f"Explore DroneBatch→{assigned} at t={current_time:.2f}")
        # flight time is max distance + service time
        t_flight = max(drone_time[van_pos, i] for i in assigned) + SERVICE_TIME
        new_del = delivered | set(assigned)
        search(van_pos, new_del, current_time + t_flight,
               plan + [f"DroneBatch→{assigned}"], van_time, drone_time, locations)
```

Fig. 3.3. Code snippet showing Branch-and-Bound search with van and drone branching (Source: program developed for this study).

D. Output and Logging

The program outputs the optimal delivery plan as a sequence of van and drone actions. The total delivery time and the number of nodes explored during the search are reported. Throughout execution, logs are printed to indicate when nodes are explored, when pruning occurs, and when new best solutions are found. This information assists in analyzing the algorithm's performance and verifying its correctness.

```python
# — Main ————————————————————————————————————————————
if __name__ == '__main__':
    locs, dist = load_data('Van_Urban_40.xlsx')
    van_time = make_time_matrix(dist, VAN_SPEED_KMH)
    drone_time = make_time_matrix(dist, DRONE_SPEED_KMH)
    # baseline
    base = greedy_van(locs, van_time)
    print(f"Baseline time: {base:.2f} min\n")
    # search optimal
    search(0, set(), 0.0, [], van_time, drone_time, locs)
    # results
    print(f"Best time: {best_time:.2f} min")
    print(f"Plan: {best_plan}")
```

Fig. 3.4. Code snippet showing main program execution and result display (Source: program developed for this study).

IV. Testing and Results

The Branch-and-Bound algorithm was tested on various problem instances to evaluate its performance in solving the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC). The tests were conducted using distance data stored in an Excel matrix, designed to reflect realistic urban delivery scenarios. Different parameter configurations were used in the experiments, including varying numbers of client locations, van and drone speeds, drone capacity, and drone maximum distance.

A. Test Scenarios and Configuration

Three test cases were prepared, with varying numbers of client locations. The following constraints were applied uniformly across all scenarios:

- Maximum number of client locations: 10
- Maximum number of drones available: 3
- Maximum drone capacity per mission: 2 deliveries
- Maximum drone reach: 1.5 km

The speed of the van and drones, as well as the service time, were configured as follows unless otherwise stated:

- Van speed: 40 km/h
- Drone speed: 60 km/h
- Service time per delivery: 0.5 minutes

The test scenarios included

1. Test Case 1: 6 client locations
2. Test Case 2: 8 client locations
3. Test Case 3: 10 client locations

These cases were selected to represent small to medium-sized instances suitable for exact solution using Branch-and-Bound.

B. Results

The Branch-and-Bound algorithm produced the optimal delivery plan for each of the three test cases while satisfying all operational constraints. The plans minimized total delivery

time by combining van movements and drone batches efficiently.

For Test Case 1 (6 client locations), the following optimal plan was generated:

- Plan: [‘Van→Client 2’, ‘DroneBatch→[1, 3, 4, 5]’]
- Total delivery time: 3.12 minutes

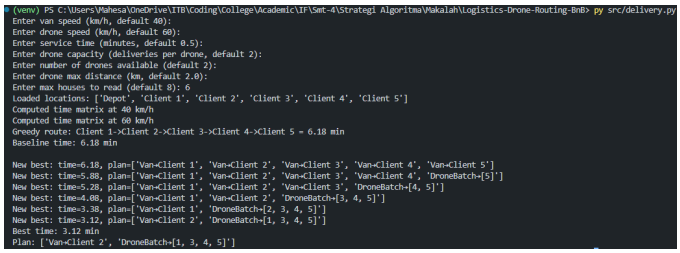


Fig. 4.1. Snapshot of program output showing optimal plan for Test Case 1 (Source: program developed for this study).

For Test Case 2 (8 client locations), the optimal plan was:

- Plan: [‘DroneBatch→[1, 2, 3, 4]’, ‘DroneBatch→[5, 6, 7]’]
- Total delivery time: 4.30 minutes

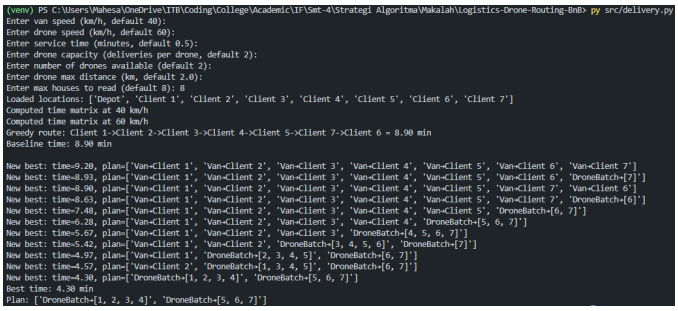


Fig. 4.2. Snapshot of program output showing optimal plan for Test Case 2 (Source: program developed for this study).

For Test Case 3 (10 client locations), the optimal plan found was:

- Plan: [‘Van→Client 2’, ‘DroneBatch→[1, 3, 4, 5]’, ‘DroneBatch→[6, 7, 8, 9]’]
- Total delivery time: 5.03 minutes

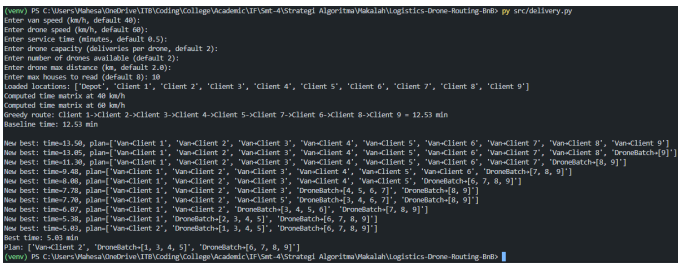


Fig. 4.3. Snapshot of program output showing optimal plan for Test Case 3 (Source: program developed for this study).

C. Discussion

The results from the three test cases demonstrated that the Branch-and-Bound algorithm was effective in determining the optimal delivery plan for the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC). The inclusion of both van deliveries and drone batch operations in the branching process provided flexibility in constructing efficient delivery sequences. The pruning mechanism, supported by the initial greedy baseline, significantly reduced the number of nodes that needed to be explored in the state-space tree.

In Test Case 1, where six clients were considered, the algorithm identified that a combination of one van delivery followed by a single drone batch covering the remaining clients resulted in the minimum delivery time. In Test Case 2, involving eight clients, the algorithm produced a solution consisting entirely of drone batches, as this strategy was more time-efficient under the given constraints. For Test Case 3, which included ten clients, the optimal plan consisted of a van delivery followed by two drone batches, highlighting the advantage of hybrid operations in larger instances.

It was observed that as the number of clients increased, the structure of the optimal plan became more complex, and the state-space tree expanded significantly. However, the Branch-and-Bound algorithm was able to compute the optimal plan in all cases within practical computation time due to effective pruning. The use of the greedy baseline as an initial upper bound was found to be beneficial, as it allowed the search to focus on promising branches early in the process.

These findings suggest that the Branch-and-Bound approach is suitable for solving VDRPMDPC instances with small to medium numbers of clients. For larger instances, additional improvements, such as tighter bounding functions or heuristic enhancements, may be required to maintain efficiency.

V. Conclusion

In this study, the Van-Drone Routing Problem with Multiple Delivery Points and Capacity Constraints (VDRPMDPC) was solved using the Branch-and-Bound algorithm. The algorithm was designed to find the optimal delivery plan that minimizes total delivery time while satisfying all operational constraints, including drone capacity, maximum drone distance, and service time at each delivery point. A greedy van-only baseline was used as the initial upper bound to enhance pruning efficiency during the search process.

The experimental results on small to medium-sized instances, consisting of up to ten client locations, demonstrated that the Branch-and-Bound algorithm was effective in producing optimal solutions. The combination of van deliveries and drone batch operations enabled the construction of delivery plans that made full use of the strengths of both delivery modes. The number of nodes explored in each test case was significantly reduced due to the pruning guided by the initial greedy baseline.

It was observed that as the number of delivery points increased, the solution space expanded rapidly, as expected for NP-hard problems. The method remained practical for instances involving up to ten clients, but larger instances would likely require additional enhancements, such as tighter bounding functions or heuristic guidance during branching, to maintain efficiency.

Overall, the Branch-and-Bound algorithm has been shown to be a suitable exact method for solving the VDRPMDPC in small to medium-scale delivery scenarios. Future work may focus on integrating real geographic data, additional constraints such as drone energy limits, or hybrid methods that combine exact algorithms with heuristics to extend the approach to larger problem instances.

VI. Appendix

The complete source code for the implementation described in this paper can be accessed at the following GitHub repository:

https://github.com/mahesa005/Logistics-Drone-Routing-BnB

The complete demonstration of the program can be accessed at the link below

https://youtu.be/6JYYVBciXQE

VII. Acknowledgment

All praise and gratitude are offered to the presence of the Almighty God, Allah Subhanahu wa Ta'ala, who has provided health, guidance, and the opportunity to complete this paper entitled "Branch-and-Bound for Van-Drone Collaborative Routing: Experiments on Real-World VDRPMDPC Instances." The author would also like to express sincere appreciation to Dr. Rinaldi Munir and Monterico Adrian, S.T., M.T., as lecturers of the course, for providing the knowledge and material that formed the foundation for this paper.

References

[1] R. Munir, Branch-and-Bound Algorithm Part 1, lecture slides, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2025.

[2] R. Munir, Branch-and-Bound Algorithm Part 2, lecture slides, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2025.

[3] R. Munir, Branch-and-Bound Algorithm Part 3, lecture slides, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2025.

[4] R. Munir, Branch-and-Bound Algorithm Part 4, lecture slides, School of Electrical Engineering and Informatics, Institut Teknologi Bandung, 2025.

[5] A. Levitin, Introduction to the Design and Analysis of Algorithms, 3rd ed. Boston: Pearson, 2012.

[6] Mendeley Data, "Van-drone collaborative routing data set," [Online]. Available: https://data.mendeley.com/datasets/4ghmjh9fpg/2/files/000078522-1fc2-4c57-a6ee-d91c643af667

[7] Pandas Development Team, "pandas 1.3.0 documentation." [Online]. Available: https://pandas.pydata.org/docs/

Statement of Originality

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

Bandung, 24 Juni 2025

Mahesa Fadhillah Andre
13523140