A Branch-and-Price Algorithm for the Vehicle Routing Problem with Drones

Hang Zhou, Hu Qin
School of Management, Huazhong University of Science and Technology, Wuhan 430074, China
{zh20010728@126.com, tigerqin1980@qq.com}

Chun Cheng*
Institute of Supply Chain Analytics, Dongbei University of Finance and Economics, Dalian 116025, China
*Corresponding author. chun.cheng@polymtl.ca

Louis-Martin Rousseau
Polytechnique Montréal and CIRRELT, Montreal H3C 3A7, Canada
louis-martin.rousseau@cirrelt.net

This paper considers a new variant of the vehicle routing problem with drones (VRPD), where multiple vehicles and drones work collaboratively to serve customers. Several practical constraints such as customers’ delivery deadlines and drones’ energy capacity are considered. Different from existing studies, we treat the number of drones taken by each vehicle as a decision variable instead of a given parameter, which provides more flexibility for planning vehicle and drone routes. We also allow a drone to perform multiple back-and-forth trips when its paired vehicle stops at a customer node. We first formulate this problem as a mixed-integer linear programming model, which is solvable by off-the-shelf commercial solvers. To tackle VRPD instances more efficiently, we next develop a set-partitioning model. To solve it, a branch-and-price algorithm is proposed, where a bidirectional labeling algorithm is used to solve the pricing problem. To speed up the algorithm, the cheapest insertion heuristic is developed for initial column generation, and a tabu search algorithm is first applied before the exact labeling algorithm for finding desired columns in each iteration of the column generation process. Extensive numerical tests show that our algorithm can solve most instances within 25 customers to optimality in a short time frame and some instances of 35 customers to optimality within a three-hour time limit. Results also demonstrate that the allocation decisions of drones can help save the duration of all routes by 3.45% on average for 25-customer instances, compared to the case of fixing the number of paired drones on each vehicle. In addition, sensitivity analyses show that multiple strategies, e.g., adopting batteries of a higher energy density and developing faster drones, can be applied to further improve the delivery efficiency of a truck-drone system.

Key words: truck-drone system; vehicle routing problem with drones; branch-and-price

1. Introduction

The development of technologies such as lighter carbon fibers and batteries of a higher energy density makes it possible to use unmanned aerial vehicles (UAVs) or drones in civil applications, such as agriculture, transportation, security, telecommunication, and media (Otto et al. 2018). Notably, drones can work in extreme environments like the aftermath of fires and earthquakes to
take over humans’ dangerous jobs. The application of drones for last-mile delivery has recently received considerable attention from industrial and academic communities. Leading companies like Amazon, Google, SF Technology, and Alibaba have designed and tested their drone-delivery modes for several years. There is also a growing body of academic research on drone-related routing problems in recent years (Poikonen and Campbell 2021).

Compared to truck delivery, drone delivery is faster, cheaper, and more flexible. Whereas drone delivery also has limitations, such as restricted payload and flight range. On the other hand, trucks often have a long travel range and can carry many parcels (Agatz et al. 2018). To leverage their complementary advantages, many researchers propose to let trucks and drones work together, forming a truck-drone delivery system. That is, trucks not only serve customers but also act as temporary hubs to launch and retrieve drones. Several companies have already adopted this innovative concept, like Mercedes-Benz (Mercedes-Benz 2017) and Amazon (Stonor 2021).

Although the truck-drone delivery mode has more strengths than the truck-only delivery mode, this application has more operational challenges. Even for a system with only a single truck and a single drone, the problem involves two types of decisions—assignment and routing. Specifically, we need to assign customers to the drone or the truck to get service and plan the visiting sequence of customers assigned to the truck.

Traditional last-mile deliveries performed by trucks are modeled as a vehicle routing problem (VRP), which has been extensively studied in the literature (Braekers et al. 2016). Whereas the routing problem with trucks and drones is only recently proposed. The first study on truck-drone delivery is by Murray and Chu (2015), which introduces the flying sidekick traveling salesman problem (FSTSP), where one truck carries one drone to serve customers. When the truck is serving a customer, the drone is set out, delivering a parcel for a single customer, then the truck and the drone meet at a new customer location. When only one truck carries one drone to deliver packages, the related problem is often called the FSTSP or the traveling salesman problem with drone (TSP-D). When multiple trucks and drones are involved, the related problem is usually called the vehicle routing problem with drones (VRPD) (Wang et al. 2017, Poikonen et al. 2017, Wang and Sheu 2019). In different studies, the VRPD has different features. For example, some studies allow trucks to wait at customers to retrieve all dispatched drones, while others impose that trucks must move to new locations to retrieve drones.

**Our Contributions.** Existing research on the VRPD often assumes that the number of drones taken by a vehicle is a given parameter instead of a decision variable, resulting in less flexibility for allocating drones and thus planning vehicle and drone routes. Moreover, heuristic algorithms are often used in the literature to solve the VRPD, and exact algorithms are seldom developed, limiting algorithm evaluation. To fill some gaps in this area, this paper introduces a new variant
of the VRPD, where multiple vehicles and drones work collaboratively to serve customers. To solve VRPD instances exactly and efficiently, we develop a branch-and-price (B&P) algorithm. Our work makes the following contributions to the literature:

1. We introduce a new variant of the VRPD, where practical constraints such as customers’ service deadlines and drones’ energy capacity are considered. Moreover, we treat the number of drones taken by each vehicle as a decision variable instead of a given parameter. To the best of our knowledge, this work is the first to consider the allocation decisions of drones to vehicles.

2. We construct a mixed-integer linear programming (MILP) model and a set-partitioning model for the problem. An exact B&P algorithm is proposed to solve the set-partitioning model, where a tailored bidirectional labeling algorithm is developed for the pricing problem. To accelerate the resolution of the pricing problem, a tabu search (TS) heuristic is first used to find desired routes before applying the exact labeling algorithm.

3. Numerical tests based on two types of benchmark instances are conducted to evaluate the performance of our B&P algorithm and the value of drone allocation decisions. Managerial insights are drawn from sensitivity analyses of key parameters.

The rest of this paper is organized as follows. Section 2 reviews related studies. Section 3 introduces the VRPD and constructs an MILP model. Section 4 presents the solution method. Section 5 describes the instance sets and provides the numerical results. Section 6 concludes this paper.

2. Literature Review

Since the introduction of the truck-drone delivery system by Murray and Chu (2015), related studies have increased rapidly. However, most works focus on heuristic solution methods, and contributions on exact algorithms are scarce. In this section, we mainly review exact solution methods for the truck-drone delivery problem, which are also summarized in Table 1. For more details about the drone-aided routing problems, see the review paper by Macrina et al. (2020).

2.1. Truck-drone Delivery System

Murray and Chu (2015) is the first to study a truck-drone delivery problem, i.e., the FSTSP, which is solved by a greedy construction heuristic algorithm. Subsequently, various variant problems are proposed, among which is the TSP-D. Slightly different from the FSTSP, the truck is allowed to wait at the launching node for the drone to return in the TSP-D (Agatz et al. 2018). Bouman et al. (2018) develop a dynamic programming method for the TSP-D, which can solve some 20-node instances to optimality within 12 hours. The authors note that the computing time can be significantly reduced if they limit the number of nodes that the vehicle can visit when the drone is away. At the same time, this limitation slightly impacts the solution quality. Poikonen et al. (2019) design a branch-and-bound (B&B) approach for the TSP-D, which can solve instances
with 20 nodes to optimality. Vásquez et al. (2021) construct a two-stage mixed-integer programming model for the TSP-D. In the first stage, they design the truck route visiting a subset of customers. The second stage optimizes the drone plan, taking the first-stage problem’s results as inputs. Based on the structure of their model, a Benders-type decomposition algorithm is developed. El-Adle et al. (2021) propose an MILP model, which is enhanced by a series of bound improvement strategies. Roberti and Ruthmair (2021) construct a compact formulation for the TSP-D and develop a B&P algorithm, which can solve instances up to 39 customers to optimality. Kang and Lee (2021) propose a heterogeneous drone-truck routing problem, where a truck armed with a heterogeneous fleet of drones is used to serve customers. When the truck stops at a customer node, drones are sent out to serve assigned customers. The truck can move to the next customer only after all drones are back. The authors develop a branch-and-cut (B&C) algorithm based on the logic-based Benders decomposition approach, which can solve 50-customer instances to optimality. Cavani et al. (2021) consider the traveling salesman problem with multiple drones. A B&C algorithm is proposed, which can solve instances with 25 customers and 3 drones to optimality.

Tamke and Buscher (2021) consider a VRPD, where each vehicle carries a fixed number of drones, starting from the depot to deliver parcels. The flight range of drones is set as a fixed distance. An MILP model with two different time-oriented objective functions is constructed. A B&C algorithm is developed, which can solve instances with 30 nodes to optimality. Wang and Sheu (2019) propose another variant of the VRPD, which is different from the most truck-drone systems considered in the literature. They assume that drones can have multiple times of flying and landing, each of which can be associated with a different truck. Moreover, drones can only land at some specific docking hubs. Trucks can visit docking hubs to carry drones during their visits to customers. The authors develop a B&P algorithm that can solve instances of up to 13 customers and 2 docking hubs to optimality. Bakir and Tinič (2020) propose a VRP with flexible drones. The authors assume that drones can be launched and retrieved at different vehicles, i.e., drones are flexible. Since the truck-drone synchronizations may happen in some already visited customer locations, they allow customers to be visited multiple times. A dynamic discretization discovery approach is introduced, which can successfully solve instances with 25 nodes to optimality. Chen et al. (2021a) and Chen et al. (2021b) propose a VRP with time windows and delivery robots, which is a variant of the VRPD. However, they assume that each robot can only deliver once, carrying one parcel, when a truck stops at a location. Our work relaxes this assumption and allows each drone to perform multiple back-and-forth trips. Moreover, an adaptive large neighborhood search (ALNS) method and a two-stage matheuristic algorithm are developed in Chen et al. (2021a) and Chen et al. (2021b), respectively.
Since a packed battery powers the flying of a drone, its flight range is closely related to energy consumption, which can be affected by several factors such as payload and weather conditions. From Table 1, we observe that most works simplify drones’ flight range as a fixed distance/duration or ignore it. These settings may lead to failures of drone routes due to energy disruptions. In our work, we explicitly consider drones’ energy capacity constraints using the battery power function in Dorling et al. (2016) and Cheng et al. (2020). Moreover, to the best of our knowledge, all existing studies do not consider the allocation decisions of drones. Instead, we optimize these decisions to better plan truck and drone routes.

2.2. Two-echelon Vehicle Routing Problem

The two-echelon vehicle routing problem (2E-VRP) is one of the closest problems to the VRPD. In the 2E-VRP, parcels from a central depot are delivered to customers through intermediate depots, called satellites. The first echelon requires a design of routes for a fleet of vehicles located at the depot to transport parcels to some satellites. The second echelon requires planning routes for vehicles located at the satellites to serve customers (Qin et al. 2021). The vehicles at the first and second echelons are similar to the trucks and drones in the VRPD, respectively. However, the vehicles in the two problems do have some different characteristics. Specifically, in the 2E-VRP, the second-echelon vehicles wait at the satellites, i.e., they are not movable from one satellite to another. In contrast, drones in the VRPD can be carried by trucks from one dispatching location to another. As the number of satellites in the 2E-VRP is often small, the first-echelon routes can be enumerated or obtained by solving a traveling salesman problem. Whereas, it is impractical to enumerate all the routes for trucks in the VRPD as they may need to visit many customers. Exact algorithms such as B&P (Santos et al. 2013, Dellaert et al. 2019) and branch-and-cut-and-price (Santos et al. 2015, Marques et al. 2020) have been designed for the 2E-VRP.

<table>
<thead>
<tr>
<th>Authors</th>
<th>#T</th>
<th>#D</th>
<th>Flight range</th>
<th>Drone capacity</th>
<th>Solution method</th>
<th>#O</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bouman et al. (2018)</td>
<td>1</td>
<td>1</td>
<td>Ignored</td>
<td>1</td>
<td>Dynamic programming</td>
<td>20</td>
</tr>
<tr>
<td>Poikonen et al. (2019)</td>
<td>1</td>
<td>1</td>
<td>Duration</td>
<td>1</td>
<td>Branch-and-bound</td>
<td>20</td>
</tr>
<tr>
<td>Vásquez et al. (2021)</td>
<td>1</td>
<td>1</td>
<td>Duration</td>
<td>1</td>
<td>Benders decomposition</td>
<td>20</td>
</tr>
<tr>
<td>El-Adle et al. (2021)</td>
<td>1</td>
<td>1</td>
<td>Duration</td>
<td>1</td>
<td>MILP and commercial solver</td>
<td>24</td>
</tr>
<tr>
<td>Roberti and Ruthmair (2021)</td>
<td>1</td>
<td>1</td>
<td>Energy related</td>
<td>1</td>
<td>Branch-and-price</td>
<td>39</td>
</tr>
<tr>
<td>Kang and Lee (2021)</td>
<td>1</td>
<td>m</td>
<td>Energy related</td>
<td>1</td>
<td>Branch-and-cut</td>
<td>50</td>
</tr>
<tr>
<td>Cavani et al. (2021)</td>
<td>1</td>
<td>m</td>
<td>Ignored</td>
<td>1</td>
<td>Branch-and-cut</td>
<td>25</td>
</tr>
<tr>
<td>Tamke and Buscher (2021)</td>
<td>m</td>
<td>m</td>
<td>Distance</td>
<td>1</td>
<td>Branch-and-cut</td>
<td>30</td>
</tr>
<tr>
<td>Wang and Sheu (2019)</td>
<td>m</td>
<td>m</td>
<td>Duration</td>
<td>m</td>
<td>Branch-and-price</td>
<td>15</td>
</tr>
<tr>
<td>Bakir and Tiniç (2020)</td>
<td>m</td>
<td>m</td>
<td>Duration</td>
<td>1</td>
<td>Dynamic discretization discovery</td>
<td>25</td>
</tr>
<tr>
<td>This paper</td>
<td>m</td>
<td>m</td>
<td>Energy related</td>
<td>1</td>
<td>Branch-and-price</td>
<td>35</td>
</tr>
</tbody>
</table>

#T: number of trucks; #D: number of drones; #O: the instance size that can be solved to optimality; m: multiple.
The most relevant work to ours is by Li et al. (2020), which introduces the 2E-VRP with time windows and mobile satellites. A homogeneous fleet of truck-drone combinations is used to deliver parcels. The term homogeneous means that the number of drones carried by each truck is identical and that the characters of each truck/drone are the same. There are three types of routes in their problem. The first is called the pure drone route; namely, the route originates and ends at the depot and is performed by a drone (i.e., the associated truck is not used). The second is called the pure truck route, i.e., the truck travels with drones not being in use. The third is called a combination route, i.e., the truck and drones work together to deliver parcels as in our problem. An ALNS method is developed to solve the problem. For more details about the 2E-VRP, see the review paper by Li et al. (2021).

3. Problem Definition and An MILP Model

In this section, we define the problem and model it as an MILP.

3.1. Problem Definition

Our problem is defined on a directed graph \( G = (V, A) \), where \( V = \{0, 1, \ldots, n\} \) is the set of nodes and \( A = \{(i, j) \mid i, j \in V, i \neq j\} \) is the set of arcs. We denote \( Z = V \setminus \{0\} \) as the set of customers. Node 0 is the central depot, denoting all feasible vehicle routes’ starting and ending nodes. Each customer has a demand \( q_i \) and a deadline \( l_i \). Parameter \( l_i \) represents the latest time to start the service at customer \( i \). \( l_0 \) is the deadline of the depot, which also denotes the planning horizon. Since some customers’ demands may surpass a drone’s load capacity or signatures are needed when delivering some parcels, not all customers are feasible to be served by a drone. Thus, we define a binary parameter \( f^d_i \) for each customer. \( f^d_i = 1 \) if customer \( i \) can be served by either a vehicle or a drone, and \( f^d_i = 0 \) if customer \( i \) can only be served by a vehicle. We denote \( Z_d = \{i \mid f^d_i = 1, i \in Z\} \) as the set of customers that can be served by drones. A homogeneous fleet of vehicles \( K_v = \{1, 2, \ldots, k_v\} \) and a homogeneous fleet of drones \( K_d = \{1, 2, \ldots, k_d\} \) are ready at the depot to deliver parcels. The capacity of vehicles is \( Q \), and each vehicle can carry at most \( k_0 \) \((k_0 \leq k_d)\) drones. Let \( q^d_i \) denote the weight of a drone and its support equipment (such as battery pack and landing platform) when installed into a vehicle. The distance between customers \( i \) and \( j \) is \( d_{ij} \), which satisfies the triangle inequality. We denote the speeds of vehicles and drones as \( vel^v \) and \( vel^d \), respectively. In general, drones travel faster than vehicles (see Wang and Sheu (2019), Otto et al. (2018), and Chung et al. (2020)), thus we assume \( vel^v \leq vel^d \). Then the traveling times of a vehicle and a drone on arc \((i, j)\) are calculated as \( t_{ij}^v = d_{ij} / vel^v \) and \( t_{ij}^d = d_{ij} / vel^d \), respectively. For a customer, vehicle and drone deliveries require different service times \( s_{ij}^v \) and \( s_{ij}^d \), respectively.
Leishman (2006) denote the power consumption of a single rotor helicopter in hover as a convex function of payload, based on which Dorling et al. (2016) derive the power consumption of a $h$-rotor drone as

$$P(q) = (W + m + q) \frac{g^3}{2 \rho \zeta R^4}$$

(1)

where $W$ is the frame weight (kg), $m$ is the battery weight (kg), $q$ is the payload (kg), $g$ is the force due to gravity (N), $\rho$ is the fluid density of air (kg/m$^3$), and $\zeta$ is the area of spinning blade disc (m$^2$). The unit of $P$ is Watt. We set $q_i^d = W + m$, satisfying $q_i^d \leq q_i^f$, and let $B_i$ be the battery energy capacity. Binary parameter $E_{ij}$ equals 1 if a drone launched from node $i$ can serve customer $j$ under the energy capacity constraint, i.e., $E_{ij} = 1$ if the constraint $P(q_i) t_{ij}^d + P(0) t_{ij}^d \leq B_i$ can be satisfied, $E_{ij} = 0$ otherwise.

In the rest of this paper, we use the terms vehicle node and drone node to represent a customer who is served by a vehicle and a drone, respectively. The term route denotes a completed route with one vehicle and several drones. Drone route represents the multiple trips performed by a drone when the associated vehicle stops at a vehicle node. The following assumptions are made on the operations of trucks and drones:

- Each drone can only be launched and retrieved by the same vehicle;
- A drone can only carry one parcel per trip but can be launched multiple times from a vehicle node. Each time before launching a drone, a fixed amount of time $t_0$ is required for loading a parcel and changing the drone’s battery;
- Drones can start to serve customers only when the associated vehicle stops at a customer location. The vehicle needs to wait there until all launched drones are back before moving to the next vehicle node;
- Each drone trip consumes a certain amount of energy. Each time when a drone starts a new trip, we swap it with a fully charged battery. As many existing studies (e.g., Poikonen et al. (2019) and Bakir and Tinci (2020)), we assume the number of available batteries is sufficient.

The objective is to minimize the total duration of all routes, including the travel times between vehicle nodes and the waiting times for drones to return at vehicle nodes. The waiting time of a vehicle is determined by its service time at the customer and the makespan of drone routes. Figure 1 shows an example of the VRPD solution, where vehicle nodes and drone nodes are denoted by squares and circles, respectively. There are three vehicles and four drones available at the central depot 0. Two drones are allocated to vehicle 1 and vehicle 2, respectively. The sequence $(0, 3, 7, 0)$ forms a route for vehicle 1. When vehicle 1 arrives at customer 3, drone 1-1 performs two trips (visiting customers 5 and 6), forming a drone route. When both drones 1-1 and 1-2 are back to vehicle 1, it moves to the next vehicle node 7. The sequence $(0, 1, 2, 0)$ forms the route of vehicle 3, where no drone is involved.
3.2. MILP Formulation

To formulate the VRPD, we define the following decision variables:

- $x_{ij}^k$: a binary variable that is 1 if vehicle $k \in K_v$ travels arc $(i, j) \in A$, 0 otherwise;
- $u_{ij}^d$: a binary variable that is 1 if customer $j \in Z_d$ is served by drone $d \in K_d$ dispatched from $i \in Z$, 0 otherwise;
- $y_{ij}^{kd}$: a binary variable that is 1 if node $j \in Z_d$ is the first customer served by drone $d \in K_d$ from vehicle $k \in K_v$ stopped at customer $i \in Z$, 0 otherwise;
- $z_{ij}$: a binary variable that is 1 if a drone visits customer $j \in Z_d$ after visiting customer $i \in Z_d$ from the same vehicle node, 0 otherwise;
- $\omega_i^k$: a continuous variable representing the amount of commodities delivered at customer $i \in Z$ by vehicle $k \in K_v$, including the demand of customer $i$ and all demands of customers served by drones dispatched from $i$. If customer $i$ is not visited by vehicle $k$, $\omega_i^k$ would be 0;
- $a_i$: a continuous variable representing the arrival time of a vehicle or drone at node $i \in V$;
- $\varphi_i$: a continuous variable representing the waiting time of a vehicle at customer $i \in Z$.

The MILP model for the VRPD is constructed as follows:

$$
\begin{align*}
\text{min} & \quad \sum_{(i,j) \in A} \sum_{k \in K_v} t_{ij}^k x_{ij}^k + \sum_{i \in Z} q_i \\
\text{s.t.} & \quad \sum_{i \in V} \sum_{k \in K_v} x_{ij}^k + \sum_{j \in Z_d} \sum_{d \in K_d} \sum_{k \in K_v} y_{ij}^{kd} + \sum_{i \in Z_d} z_{ij} = 1 & \forall j \in Z, \\
& \quad \sum_{j \in Z} x_{ij}^k \leq 1 & \forall k \in K_v, \\
& \quad \sum_{i \in V} \sum_{j \in Z} x_{ij}^k = \sum_{j \in Z_d} \sum_{i \in V} x_{ji}^k & \forall j \in V, k \in K_v, \\
& \quad \sum_{j \in Z_d} \sum_{d \in K_d} y_{ij}^{kd} \leq k_0 \sum_{j \in V} x_{ji}^k & \forall i \in Z, k \in K_v.
\end{align*}
$$
\[
\begin{align*}
\sum_{j \in Z_d} \sum_{k \in K_v} y_{ij}^d & \leq 1 \quad \forall i \in Z, d \in K_d, \\
\sum_{j \in Z_d} u_{ij}^d & \leq M \sum_{j \in Z_d} \sum_{k \in K_v} y_{ij}^d \quad \forall i \in Z, d \in K_d, \\
u_{ij}^d & \leq \sum_{k \in K_v} y_{ij}^d + \sum_{l \in Z_d} z_{ij}, \quad \forall i \in Z, j \in Z_d, d \in K_d, \\
u_{ij}^d & \geq \sum_{k \in K_v} y_{ij}^d \quad \forall i \in Z, j \in Z_d, d \in K_d, \\
\sum_{j \in Z_d} \sum_{k \in K_v} u_{ij}^d & \geq \sum_{j \in Z_d} z_{ij}, \quad \forall i \in Z_d, \\
u_{ij}^d - u_{ij}^d & \leq 1 - z_{ij}, \quad \forall i \in Z_d, j \in Z_d, l \in Z, d \in K_d, \\
u_{ij}^d & \leq E_{ij}, \quad \forall i \in Z, j \in Z_d, d \in K_d, \\
q_j + \sum_{i \in Z_d} \sum_{d \in K_d} q_i u_{ij}^d - \omega_j^k & \leq M_j (1 - \sum_{i \in V} x_{ij}^k), \quad \forall j \in Z, k \in K_v, \\
\sum_{j \in Z} \omega_j^k + q_i^k \sum_{d \in K_v} \sum_{j \in Z_d} y_{ij}^d & \leq Q, \quad \forall i \in Z, k \in K_v, \\
t_0^i - a_j & \leq M_j (1 - \sum_{k \in K_v} x_{0j}^k), \quad \forall j \in Z, \\
a_i + \varphi_i + t_{ij}^d - a_j & \leq M_j (1 - \sum_{k \in K_v} x_{ij}^k), \quad \forall i \in Z, j \in V, \\
a_i + t_0 + t_{ij}^d - a_j & \leq M_j (1 - \sum_{k \in K_v} \sum_{d \in K_d} y_{ij}^d), \quad \forall i \in Z, j \in Z_d, \\
a_i + s_i^d + t_{ij}^d + t_0 + t_{ij}^d - a_j & \leq M_j (2 - z_{ij} - \sum_{d \in K_d} u_{ij}^d), \quad \forall l \in Z, i \in Z_d, j \in Z_d, \\
\varphi_i & \geq s_i^d \sum_{j \in Z_v} \sum_{k \in K_v} x_{ij}^k, \quad \forall i \in Z, \\
a_i + s_i^d + t_{ij}^d - a_i - \varphi_i & \leq M_j^\prime (1 - \sum_{d \in K_d} u_{ij}^d), \quad \forall i \in Z, j \in Z_d, \\
a_i & \leq l_i, \quad \forall i \in V, \\
x_{ij}^k & \in \{0,1\}, \quad \forall (i,j) \in A, k \in K_v, \\
y_{ij}^d & \in \{0,1\}, \quad \forall i \in Z, j \in Z_d, k \in K_v, d \in K_d, \\
u_{ij}^d & \in \{0,1\}, \quad \forall i \in Z, j \in Z_d, d \in K_d, \\
z_{ij} & \in \{0,1\}, \quad \forall i \in Z_d, j \in Z_d, \\
\omega_j^k & \geq 0, \quad \forall i \in Z_d, j \in Z_d, \\
a_i & \geq 0, \quad \forall i \in V, \\
\varphi_i & \geq 0, \quad \forall i \in Z.
\end{align*}
\]

The objective function (2) minimizes the total duration of all routes. Constraints (3) ensure that each customer is visited by a vehicle or a drone exactly once. Constraints (4) mean that each vehicle can leave the depot at most once. Constraints (5) ensure the flow conservation at each node. Constraints (6) indicate that drones can be launched from customer locations only where
a vehicle stops. Constraints (7) impose that a drone starts the drone route at most once from a vehicle node. Constraints (8)–(12) denote the multi-trip property of a drone route. In particular, constraints (8) restrict that only when a drone starts a drone route at customer \( i \), \( u_{ij}^d \) can be 1. We set the large constant \( M = |Z_d| \). Constraints (9) mean that if node \( j \) is neither the first drone node nor the subsequent nodes in a drone route, then \( u_{ij}^d \) would be 0. On the contrary, constraints (10)–(12) indicate that when \( j \) is the first drone node or the subsequent nodes in a drone route, \( u_{ij}^d \) would be 1. Constraints (13) mean that we can use a drone to serve a customer only when it is energy feasible. Inequalities (14) calculate the value of \( \omega_j^k \), and we set the large constant \( M_j = q_j + \sum_{i \in Z_d} q_i \). Inequalities (15) ensure that vehicle capacity constraints are respected. The total payload of a vehicle includes the demands of customers and the weight of drones and their support equipment. Constraints (16)–(19) calculate the arrival times at customers. We set the large constants \( M_j^l = t_{ij}, M_{ij} = l_j + t_{ij}, \) and \( M_{ij}^r = l_j + t_{ij} + t_{ij}^d \). Constraints (20) and (21) calculate the waiting time at a customer location, which is the maximum between the makespan of drone routes and the vehicle’s service time at that customer. We set the large constants \( M_{ij} = l_j + s_j^d + t_{ij}^d + t_0 + t_{ij}^d \) and \( M_{ij}^r = l_j + s_j^d + t_{ij}^d \). Inequalities (22) impose the deadline constraints. Constraints (23)–(29) define the domains of all decision variables.

4. Branch-and-Price Algorithm

Although the MILP model (2)–(29) can be tackled by off-the-shelf solvers like CPLEX and Gurobi, solvers can usually not provide optimal solutions for medium- or large-size instances in a reasonable computing time, because the VRPD is NP-hard—it reduces to the VRP when \( K_d = \emptyset \), which is a well-known NP-hard problem. To solve VRPD instances more efficiently, this section introduces a B&P algorithm based on a set partitioning model.

4.1. Set Partitioning Formulation

To construct the set partitioning model, we introduce the following notation:

- \( \mathcal{R} \): the set of all feasible routes;
- \( c_r \): the shortest duration of route \( r \in \mathcal{R} \);
- \( a_i \): the number of times customer \( i \in Z \) is visited by route \( r \in \mathcal{R} \);
- \( d_r \): the number of drones allocated to route \( r \in \mathcal{R} \);
- \( \mu_r \): a binary variable equalling 1 if route \( r \in \mathcal{R} \) is selected, 0 otherwise.

A route \( r \) is feasible only if the following constraints are satisfied: (i) it starts and ends at the depot, (ii) it visits a customer exactly once, (iii) the number of drones taken by the vehicle does not exceed \( k_0 \), and (iv) the vehicle capacity constraint, drone energy capacity constraint, and customer deadline constraints are respected.
The set partitioning formulation for the VRPD is constructed as follows:

\[
\begin{align*}
\text{min} & \quad \sum_{r \in R} c_r \mu_r \\
\text{s.t.} & \quad \sum_{r \in R} \alpha_{ir} \mu_r = 1 & \forall i \in Z, \\
& \quad \sum_{r \in R} \mu_r \leq k_v & \forall r \in R, \\
& \quad \sum_{r \in R} d_r \mu_r \leq k_d & \forall r \in R, \\
& \quad \mu_r \in \{0, 1\} & \forall r \in R.
\end{align*}
\]

The objective function (30) minimizes the total duration of selected routes. Constraints (31) force each customer to be visited exactly once. Constraints (32)–(33) limit the number of available vehicles and drones, respectively. Constraints (34) define the domain of variables.

Due to the large size of set \( R \), it is impossible to enumerate all the feasible routes. Thus, we use a column generation (CG) algorithm to solve the linear relaxation problem (LRP) at each node of the B&B tree, leading to a B&P algorithm. In the CG framework, we first solve a restricted master problem (RMP) by considering the LRP with a subset \( R' \subseteq R \). After its resolution, dual solutions are passed to the pricing problem to check if there exist feasible routes with negative reduced costs. If so, these routes are added to set \( R' \), and the RMP is solved again; otherwise, the current solution can be extended to generate an optimal solution for the LRP.

4.2. Cheapest Insertion Heuristic

To accelerate the CG process, we use the cheapest insertion heuristic to generate initial columns. The pseudo-code is provided in Algorithm 1. In general, to minimize the duration, it is better to serve a customer by a drone than by a vehicle. Thus, we implement the cheapest insertion heuristic by greedily assigning as many customers as possible to be served by drones. Let \( \Phi \) be the set of unserved customers. For each \( i \in \Phi \), we define a set \( \phi(i) \), including all unserved customers reachable by a drone from \( i \). We first initialize \( k_v \) empty routes and evenly distribute drones to each route. We then sort all customers in \( \Phi \) in non-ascending order concerning the value of \( |\phi(i)| \).

Next, we treat the first node \( i' \) in \( \Phi \) as a vehicle node, which is subsequently inserted into the best feasible position that minimizes the increase of route duration. Finally, we determine drone routes for customers in \( \phi(i') \). Our objective is to minimize the makespan of drone routes, and thus we need to determine the duration of the latest completed drone route. To construct routes for drones, we first sort all nodes in \( \phi(i') \) in non-descending order concerning their service deadlines. We then insert the first node in \( \phi(i') \) to the drone route with the minimum completion time. If the insertion operation is feasible, we remove this node from \( \Phi \). The procedures described above are repeated until \( \Phi = \emptyset \).
Algorithm 1 The pseudo code of the cheapest insertion heuristic.

1: **Input**: customer set $Z$, vehicle number $k_v$, and drone number $k_d$.
2: **Define** $\Phi$: the set of unserved customers; $\phi(i)$: the set of unserved customers reachable by a drone from $i \in \Phi$.
3: **Initialize** $\Phi \leftarrow Z$; $R' \leftarrow k_v$ empty routes, each with $\lfloor k_d/k_v \rfloor$ or $\lceil k_d/k_v \rceil$ drones.
4: **while** $\Phi \neq \emptyset$ **do**
5: **for** all $i \in \Phi$ **do**
6: \[ \phi(i) \leftarrow \{ j : E_{ij} = 1, j \in \Phi \} . \]
7: **end** for
8: Sort $\Phi$ in non-ascending order concerning the value of $|\phi(i)|$.
9: \[ i' \leftarrow \text{the first node in } \Phi; \delta_{\text{min}} \leftarrow +\infty. \]
10: **for** all route $r \in R'$ **do**
11: **for** all insertion position $pos$ in route $r$ **do**
12: \[ \text{if position } pos \text{ is feasible for inserting } i' \text{ then} \]
13: \[ \text{Calculate the incremental time } \delta \text{ after inserting } i' \text{ into } pos \text{ of route } r. \]
14: \[ \text{if } \delta < \delta_{\text{min}} \text{ then} \]
15: \[ \delta_{\text{min}} = \delta, r^* = r, pos^* = pos. \]
16: **end** if
17: **end** if
18: **end** for
19: **end** for
20: Insert customer $i'$ into position $pos^*$ of route $r^*$; $\Phi \leftarrow \Phi \setminus \{i'\}$.
21: Sort $\phi(i')$ in non-descending order concerning customers’ service deadlines.
22: **for** all $j \in \phi(i')$ **do**
23: Select the drone route $dr$ with the minimum completion time.
24: **if** the end of $dr$ is feasible for inserting $j$ **then**
25: Insert $j$ into the end of route $dr$; $\Phi \leftarrow \Phi \setminus \{j\}$.
26: **end** if
27: **end** for
28: **end while
29: **Output**: route set $R'$.

4.3. The Pricing Problem

Let $\lambda_i, i \in Z$ be the dual variables associated with constraints (31), and $\lambda_0^v$ and $\lambda_0^d$ be the dual variables associated with constraints (32) and (33), respectively. The reduced cost of route $r \in R$ is

\[ \bar{c}_r = c_r - \sum_{i \in Z} a_{ir} \lambda_i - \lambda_0^v - b_r \lambda_0^d. \]  
(35)
The purpose of the pricing problem is to find routes with negative reduced costs. For the VRPD, the pricing problem is an elementary shortest path problem with drones and resource constraints (ESPPDRC). To solve the ESPPDRC, we apply a bidirectional labeling algorithm proposed by Righini and Salani (2006), consisting of extending labels in both forward and backward directions and then merging forward and backward labels to form complete feasible routes. To guarantee that all routes with a different number of carried drones are searched, we run the labeling algorithm $k_0 + 1$ times, each time with a different number of associated drones. For example, if each vehicle can carry at most $k_0 = 3$ drones, we then run the labeling algorithm four times, assuming that $0, 1, 2,$ and $3$ drones are carried, respectively. In the following sections, we use $\bar{k}_d$ to represent the number of drones carried by a vehicle in the current labeling procedure.

4.3.1. **Forward Labeling.** Let $p(L_f)$ be a feasible forward partial path denoted by a label $L_f = (v(L_f), \sigma(L_f), \pi'(L_f), \kappa(L_f), C(L_f), \Omega_1(L_f), \Omega_2(L_f))$, where the attributes are defined as follows:

- $v(L_f)$: the last node added to the partial path, which can be a vehicle node or a drone node;
- $\sigma(L_f)$: the last vehicle node visited on $p(L_f)$;
- $\pi'(L_f)$: the time of drone $i$ ($i = 1, \ldots, \bar{k}_d$) returning to $\sigma(L_f)$ after serving all assigned customers. If $i = 0$, $\pi'(L_f)$ denotes the time when the vehicle finishes serving $\sigma(L_f)$;
- $\kappa(L_f)$: the remaining vehicle capacity after serving all customers on $p(L_f)$;
- $C(L_f)$: the accumulated dual value of the partial path $p(L_f)$. Note that the terms involved $\lambda^0$ and $\lambda^d$ are not included in $C(L_f)$ and we add them in the labeling joining procedure;
- $\Omega_1(L_f)$: the set of vehicle nodes that could be reached from $\sigma(L_f)$;
- $\Omega_2(L_f)$: the set of drone nodes that could be reached from $\sigma(L_f)$ without considering the battery energy capacity constraint.

The forward labeling procedure starts from the initial label $L_f = (0, 0, 0, Q - q_0\bar{k}_d, 0, V, Z_d)$. Given a label $L'_f$, a new label $L_f$ can be created by adding a new node $w$ to the end of the partial path $p(L'_f)$. We discuss conditions for a feasible extension and derive the relations for generating $L_f$ by considering two cases: $w$ is a vehicle node or a drone node.

**Case 1.** If $w \in \Omega_1(L'_f)$, a feasible extension must satisfy $\kappa(L'_f) - q_w \geq 0$ and $\max_{j=0,\ldots,\bar{k}_d} \{ \pi'(L'_f) \} + t^v_{\sigma(L'_f)w} \leq l_w$. The following relations are applied to obtain the attributes of label $L_f$.

\begin{align*}
v(L_f) &= w \\
\sigma(L_f) &= w \\
\pi'(L_f) &= \begin{cases} 
\max_{j=0,\ldots,\bar{k}_d} \{ \pi'(L'_f) \} + t^v_{\sigma(L'_f)w} & \text{if } i = 1, \ldots, \bar{k}_d \\
\max_{j=0,\ldots,\bar{k}_d} \{ \pi'(L'_f) \} + t^v_{\sigma(L'_f)w} + s^w & \text{if } i = 0 
\end{cases} \\
\kappa(L_f) &= \kappa(L'_f) - q_w
\end{align*}

(36)  
(37)  
(38)  
(39)
CASE 2. If \( w \in \Omega_2(L'_f) \), a feasible extension must satisfy \( \kappa(L'_f) - q_w \geq 0 \) and \( E_{\sigma(L'_f)}w = 1 \), and there exists at least one index \( i \in \{1, \ldots, \tilde{k}_d\} \) such that \( \pi^i(L'_f) + t_0 + t_{\sigma(L'_f)w}^d \leq l_w \). The following relations are applied to generate label \( L_f \).

\[
\nu(L_f) = w \tag{43}
\]

\[
\sigma(L_f) = \sigma(L'_f) \tag{44}
\]

\[
\pi^i(L_f) = \begin{cases} 
\pi^i(L'_f) & \text{if } i = 0 \\
\pi^i(L'_f) + t_0 + t_{\sigma(L'_f)w}^d + s_w + t_{w\sigma(L'_f)}^d & \text{if drone } i \in \{1, \ldots, \tilde{k}_d\} \text{ visits } w
\end{cases} \tag{45}
\]

\[
\kappa(L_f) = \kappa(L'_f) - q_w \tag{46}
\]

\[
C(L_f) = C(L'_f) + \lambda_w \tag{47}
\]

\[
\Omega_1(L_f) = \Omega_1(L'_f) - \{ w' : (w, w') \in A, \max_{i=0, \ldots, \tilde{k}_d} \{ \pi^i(L_f) \} + t_{\sigma(L'_f)w}^d > l_w \text{ or } \kappa(L_f) - q_w < 0 \} - \{ w \} \tag{48}
\]

\[
\Omega_2(L_f) = \Omega_2(L'_f) - \{ w' : w' \in Z_d, \min_{i=1, \ldots, \tilde{k}_d} \{ \pi^i(L_f) \} + t_0 + t_{\sigma(L'_f)w}^d > l_w \text{ or } \kappa(L_f) - q_w < 0 \} - \{ w \}. \tag{49}
\]

If there exist multiple feasible drone routes to insert \( w \), we generate all the related labels. Thus, when extending label \( L'_f \), multiple new labels might be generated as it might be feasible to treat \( w \) as a drone node or a vehicle node. We note that the update of \( \Omega_2(L_f) \) in (42) only applies when \( vel^d \geq vel^p \), otherwise the triangle inequality of travel time \( d_{ij} / vel^d \leq d_{ik} / vel^p + \min\{s^p_k + d_{kj} / vel^p, d_{kj} / vel^d\} \) may not hold. Fortunately, the relation \( vel^d \geq vel^p \) holds in most practical applications, thus we keep this assumption in our model.

Apparently, a large number of labels will be generated and stored in the procedure of the labeling algorithm. However, some labels will not lead to an optimal route. We discard them via dominance rules. Note that in a partial route, each drone route has the same status. Thus, combining a backward partial path with \( p(L_f) \) in different orders of drone routes will create several new while completely equivalent routes. Whereas, in the labeling algorithm for traditional routing problems like the VRP, the combination of two partial paths only produces one completed route. To handle the special case in our problem, we propose an enhanced dominance rule, which takes all produced completed routes into account. Let \( P(L_f) \) be the set of all feasible backward partial paths that can be connected with \( p(L_f) \) to produce at least one feasible completed route. Let \( p^h \in P(L_f) \) be one of the partial paths and \( R(p(L_f) \oplus p^h) \) be the set of feasible completed routes obtained by
combining \( p(L_f) \) with \( p^b \). Let \( p \in \mathcal{R}(p(L_f) \oplus p^b) \) be one of the completed routes and \( \bar{C}(p) \) be the corresponding reduced cost. \( L^1 \prec L^2 \) means that label \( L^1 \) dominates \( L^2 \).

**Definition 1** For two labels \( L^1_f \) and \( L^2_f \) satisfying \( \sigma(L^1_f) = \sigma(L^2_f) \), \( L^1_f \prec L^2_f \) if

1. \( P(L^2_f) \subseteq P(L^1_f) \);
2. \( \min_{p_1 \in \mathcal{R}(p(L^1_f) \oplus p^b)} \{ \bar{C}(p_1) \} \leq \min_{p_2 \in \mathcal{R}(p(L^2_f) \oplus p^b)} \{ \bar{C}(p_2) \} \quad \forall p^b \in P(L^2_f) \).

Definition 1 states that all the partial paths that can be connected with \( L^2_f \) to form completed routes can also be connected with \( L^1_f \). Moreover, \( L^1_f \) can always create at least one completed route with a reduced cost being no more than those of all routes created by \( L^2_f \). Using this definition, we propose the following dominance rule:

**Proposition 1** \( L^1_f \prec L^2_f \) if the following conditions are satisfied

\[
\sigma(L^1_f) = \sigma(L^2_f) \tag{50}
\]

\[
\pi^i(L^1_f) \leq \pi^i(L^2_f) \quad i = 0, \ldots, \bar{k}_d \tag{51}
\]

\[
\kappa(L^1_f) \geq \kappa(L^2_f) \tag{52}
\]

\[
\Omega_1(L^2_f) \subseteq \Omega_1(L^1_f) \tag{53}
\]

\[
\Omega_2(L^2_f) \subseteq \Omega_2(L^1_f) \tag{54}
\]

\[
\pi^i(L^1_f) - C(L^1_f) - \pi^i(L^2_f) + C(L^2_f) \leq 0 \quad i = 0, \ldots, \bar{k}_d. \tag{55}
\]

This dominance rule does not require that \( v(L^1_f) = v(L^2_f) \), which extends the space for performing dominance tests. Conditions (50)–(54) ensure that \( P(L^2_f) \subseteq P(L^1_f) \) and conditions (55) guarantee the relation between the reduced costs. Note that as all drones are identical, the routes performed by different drones at \( \sigma(L_f) \) can be mutually exchanged, leading to different combinations of \( \pi^i(L_f) \). Figure 2 shows an example, where vehicle nodes and drone nodes are denoted by rectangles and circles, respectively. Dotted lines are used to represent the service orders of drone nodes. Label \( L_f \) with \( \sigma(L_f) = 1 \) has two drone routes, whose service completion times are 10 and 20, respectively. For combination 1, we have \( \pi^1(L_f) = 10 \) and \( \pi^2(L_f) = 20 \). Since drones are homogeneous, we can exchange their routes as shown in combination 2, where \( \pi^1(L_f) = 20 \) and \( \pi^2(L_f) = 10 \). These changes will not affect the waiting time of the vehicle at the customer node. Thus, when applying the dominance rule, we only need to find one combination of \( \pi^i(L_f) \) that satisfies conditions (51) and (55).

**Proof of Proposition 1**: Consider two labels \( L^1_f \) and \( L^2_f \) that satisfy conditions (50)–(55).

We first prove \( P(L^2_f) \subseteq P(L^1_f) \). For any partial route \( p^b \in P(L^2_f) \), let \( p_1 \) be one of the routes obtained by combining \( p(L^1_f) \) with \( p^b \). Condition (52) guarantees that \( p_1 \) satisfies the vehicle capacity constraint. Let \( S(p) \) be the set of nodes visited along \( p \). Conditions (53)–(54) guarantee that
Figure 2  An example of drone route exchange.

\[
S(p^k) \subseteq \left( \Omega_1(L^2_f) \cup \Omega_2(L^2_f) \right) \subseteq \left( \Omega_1(L^1_f) \cup \Omega_2(L^1_f) \right), \text{ so } S(p(L^1_f)) \cap S(p^k) = \emptyset \text{ and } p_1 \text{ is an elementary route. Condition (50) guarantee that the drone nodes visited from } \sigma(L^1_f) \text{ satisfy the battery energy constraint for route } p_1. \text{ Finally, according to condition (51), } p(L^1_f) \text{ can depart earlier at all drone routes and the vehicle route than } p(L^2_f). \text{ Due to } p^k \in P(L^2_f), \text{ there must exist a route } p_2 \in \mathcal{R}(p(L^2_f) \oplus p^k). \text{ If we connect } p(L^1_f) \text{ and } p^k \text{ with the same order as } p_2 \text{ to obtain } p_1, \text{ then } p_1 \text{ must satisfy customers’ deadlines. To summarize, for any } p^k \in P(L^2_f), \text{ we can always find a corresponding route } p_1 \in \mathcal{R}(p(L^1_f) \oplus p^k). \text{ Thus, for all } p^k \in P(L^2_f), \text{ the relation } p^k \in P(L^1_f) \text{ holds, and then we have } P(L^2_f) \subseteq P(L^1_f).

Next, we prove that for any } p^k \in P(L^2_f), p_2 \in \mathcal{R}(p(L^2_f) \oplus p^k), \text{ there exists } p_1 \in \mathcal{R}(p(L^1_f) \oplus p^k) \text{ guaranteeing the relation } \hat{C}(p_1) \leq \hat{C}(p_2) \text{ holds. We first define some attributes associated with the backward partial path } p^k. \text{ Let } \rho'(p^k) \text{ be the time duration of drone route } i \ (i = 1, \ldots, \tilde{k}_d) \text{ started from } \sigma(L^1_f). C_1(p^k) \text{ is the accumulated dual value of path } p^k. C_2(p^k) \text{ is the time duration from the last vehicle node to the end node 0. Then we have }

\[
\hat{C}(p_1) - \hat{C}(p_2) = \max \left\{ \max_{i=1,\ldots,\tilde{k}_d} \left\{ \pi'(L^1_i) + \rho'(p^k) \right\} , \pi^0(L^1_i) \right\} + \rho'(L^1_i) + C_2(p^k) - C(L^1_i) + C_1(p^k) \\
- \max \left\{ \max_{i=1,\ldots,\tilde{k}_d} \left\{ \pi'(L^2_i) + \rho'(p^k) \right\} , \pi^0(L^1_i) \right\} - \rho'(L^1_i) - C_2(p^k) + C(L^2_i) + C_1(p^k) \\
= \max \left\{ \max_{i=1,\ldots,\tilde{k}_d} \left\{ \pi'(L^1_i) + \rho'(p^k) - C(L^1_i) \right\} , \pi^0(L^1_i) - C(L^1_i) \right\} \\
- \max \left\{ \max_{i=1,\ldots,\tilde{k}_d} \left\{ \pi'(L^2_i) + \rho'(p^k) - C(L^2_i) \right\} , \pi^0(L^1_i) - C(L^1_i) \right\} \\
\leq \max \left\{ \max_{i=1,\ldots,\tilde{k}_d} \left\{ \pi'(L^1_i) + \rho'(p^k) - C(L^1_i) \right\} - \max_{i=1,\ldots,\tilde{k}_d} \left\{ \pi'(L^2_i) + \rho'(p^k) - C(L^2_i) \right\} , \pi^0(L^1_i) - C(L^1_i) - \pi^0(L^2_i) + C(L^2_i) \right\} \\
\leq \max \left\{ \max_{i=1,\ldots,\tilde{k}_d} \left\{ \pi'(L^1_i) - C(L^1_i) - \pi'(L^2_i) + C(L^2_i) \right\} , \pi^0(L^1_i) - C(L^1_i) - \pi^0(L^2_i) + C(L^2_i) \right\}
\]
\[
\begin{align*}
\max_{i=0, \ldots, \bar{k}_d} \left\{ \pi'(L^1_i) - C(L^1_i) - \pi'(L^2_i) + C(L^2_i) \right\} \\
\leq 0.
\end{align*}
\]

The third and fourth inequalities are obtained based on Theorem 1. As there may exist multiple combinations of \(\pi'(L_i)\) satisfying constraints (51) and (55), we only need to find one combination to guarantee the relations established. \(\square\)

**Theorem 1** For any \(a_i \in \mathbb{R}, b_i \in \mathbb{R}, i = 1, \ldots, \bar{k}_d\), the following relation holds

\[
\max_{i=1, \ldots, \bar{k}_d} \{a_i\} - \max_{i=1, \ldots, \bar{k}_d} \{b_i\} \leq \max_{i=1, \ldots, \bar{k}_d} \{a_i - b_i\}.
\]

**Proof of Theorem 1:** Let \(\max_{i=1, \ldots, \bar{k}_d} \{a_i\} = a_{i_1}\) and \(\max_{i=1, \ldots, \bar{k}_d} \{b_i\} = b_{i_2}\), then we have

\[
\max_{i=1, \ldots, \bar{k}_d} \{a_i\} - \max_{i=1, \ldots, \bar{k}_d} \{b_i\} = a_{i_1} - b_{i_2} \leq a_{i_1} - b_{i_1} \leq \max_{i=1, \ldots, \bar{k}_d} \{a_i - b_i\}. \quad \square
\]

To check if relations (51) and (55) are satisfied, we may need to compare (at most) \(\bar{k}_d!\) times for all combinations of \(\pi'(L^1_i)\) and \(\pi'(L^2_i)\). As both relations involve the calculation of \(\pi'(L^1_i) - \pi'(L^2_i)\) for all \(i = 0, \ldots, \bar{k}_d\), under given combinations we only need to find the value of \(\max_{i=0, \ldots, \bar{k}_d} \{\pi'(L^1_i) - \pi'(L^2_i)\}\) and check if it satisfies the relations. If the minimum of \(\max_{i=0, \ldots, \bar{k}_d} \{\pi'(L^1_i) - \pi'(L^2_i)\}\) satisfy conditions (51) and (55) and all other conditions are also met, we can say label \(L^1_i < L^2_j\). To help understand this statement, an example is given as follows. We assume that two labels \(L^1_i\) and \(L^2_j\) satisfy constraints (50), (52)–(54) and \(\bar{k}_d = 3\) drones are used. Meanwhile, \(\pi'(L^1_1) = 15, \pi'(L^1_2) = 10, \pi'(L^1_3) = 7\) and \(\pi'(L^2_1) = 9, \pi'(L^2_2) = 13, \pi'(L^2_3) = 17\). Note that this is only one combination of \(\pi'(L^1_i)\) and \(\pi'(L^2_i)\), and each of them has a total of \(\bar{k}_d! = 3! = 6\) combinations. Constraints (51) and (55) can be converted to \(\max_{i=0, \ldots, \bar{k}_d} \{\pi'(L^1_i) - \pi'(L^2_i)\}\) \(\leq \max\{0, C(L^1_i) - C(L^2_i)\}\). For the given combinations, \(\max_{i=0, \ldots, \bar{k}_d} \{\pi'(L^1_i) - \pi'(L^2_i)\}\) = 6 > 0, thus they do not meet the dominance rule. However, if we exchange \(\pi'(L^1_1)\) with \(\pi^3(L^1_1)\), i.e., let \(\pi'(L^1_1) = 7\) and \(\pi'(L^1_2) = 15\), we will have a smaller value of \(\max_{i=0, \ldots, \bar{k}_d} \{\pi'(L^1_i) - \pi'(L^2_i)\}\), which equals \(\pi^2(L^1_1) - \pi^2(L^2_1)\) = \(-3 < 0\), satisfying the constrains. The problem is then transformed to find one pair of combinations of \(\pi'(L^1_i)\) and \(\pi'(L^2_j)\) minimizing \(\max_{i=0, \ldots, \bar{k}_d} \{\pi'(L^1_i) - \pi'(L^2_i)\}\).

**Corollary 1** The minimum of \(\max_{i=1, \ldots, \bar{k}_d} \{\pi'(L^1_i) - \pi'(L^2_i)\}\) can be obtained when \(\pi'(L^1_i)\) and \(\pi'(L^2_j)\) are sorted in the same order (e.g., both are in non-ascending or non-descending order).

**Proof of Corollary 1:** We prove it by contradiction. Assume we have two series of numbers, \(a_1, a_2, \ldots, a_n\) and \(b_1, b_2, \ldots, b_n\) which are both sorted in non-descending order. We denote \(b_1, b_2, \ldots, b_n\) as order A. Assume there exists an order \(b_{k_1}, b_{k_2}, \ldots, b_{k_n}\) satisfying \(\max_{i=1, \ldots, n} \{a_i - b_i\} < \max_{i=1, \ldots, n} \{a_i - b_i\}\), which is denoted as order B. Let \(i' = \arg\max_{i=1, \ldots, n} \{a_i - b_i\}\) and \(j' = \arg\max_{i=1, \ldots, n} \{a_i - b_i\}\).
argmax_{j=1,...,n} \{a_j - b_{k_j}\}. According to the assumption \(a_{i^*} - b_{k_{i^*}} < a_{i'} - b_{k_{i'}}\), in order \(B\) there must exist an index \(k'_{i^*} \in \{i' + 1, i' + 2, \ldots, n\}\), otherwise the relation \(a_{i^*} - b_{k_{i^*}} \geq a_{i'} - b_{k_{i'}}\) will be violated. Similarly, \(k_{i^*+1}, k_{i^*+2}, \ldots, k_n\) should all belong to \(\{i' + 1, i' + 2, \ldots, n\}\). If \(k'_{i^*} (i^* > i')\) does not belong to \(\{i' + 1, i' + 2, \ldots, n\}\), then there exists the relation \(a_{i'} - b_{k_{i'}} \geq a_{i^*} - b_{k_{i^*}} > a_{i^*} - b_{k'_{i^*}}\). However, only \(n - i'\) elements are in this set, which makes it impossible to construct a bijection. \(\square\)

Using Corollary 1, we can sort \(\pi'(L_f)\) and \(\pi'(L_f')\) before applying dominance rules to reduce the time complexity from \(O(n!)\) to \(O(1)\). As mentioned before, when a label is extended to a drone node, it might be feasible for each carried drone to visit this node, creating multiple new labels. The following proposition can help reduce the number of new labels without sacrificing the optimality of the partial route.

**Proposition 2** When a forward label \(L_f\) is extended to a drone node, we only need to consider the case where drone \(i' = \text{argmin}_{i=1,...,k_d} \{\pi'(L_f)\}\) visits this new node, which will not affect the optimality of the partial route.

Before proving Proposition 2, we give the following definitions:

**Definition 2** *(Final optimal)* Suppose we have two labels \(L_1^f\) and \(L_2^f\) satisfying (i) \(\sigma(L_1^f) = \sigma(L_2^f) = \sigma(L_f)\), (ii) the partial routes before reaching \(\sigma(L_f)\) are the same, and (iii) the drone nodes to be served from \(\sigma(L_f)\) are also the same. Whereas, the drone routes from \(\sigma(L_f)\) are different. If \(\max_{i=0,...,k_d} \{\pi'(L_1^f)\} \leq \max_{i=0,...,k_d} \{\pi'(L_2^f)\}\), we call \(L_1^f\) a final optimal label. It means that for all drone nodes to be served from \(\sigma(L_f)\), label \(L_1^f\) must be one of the best labels when we are extending a vehicle node.

**Definition 3** *(The same type of final optimal)* For label \(L_1^f\) and \(L_2^f\), if (i) \(\sigma(L_1^f) = \sigma(L_2^f) = \sigma(L_f)\), (ii) the partial routes before reaching \(\sigma(L_f)\) are same, (iii) the drone nodes to be served from \(\sigma(L_f)\) are the same, and (iv) they are all final optimal labels, then we call these two labels the same type of final optimal labels.

**Definition 4** *(Final optimal*) A final optimal label with the following property is called a final optimal label: Removing the last drone node in any drone route, the completion time of this route will be less than or equal to those of all other drone routes.

**Proof of Proposition 2:** The new label extended based on Proposition 2 cannot dominate other new labels extended using the original method (i.e., considering all feasible drone routes), but can obtain all final optimal labels. And there must exist at least one final optimal label in each type of final optimal labels.

We first prove, for each type of final optimal labels, there must exist at least one feasible final optimal label. We assume there exists a feasible label \(L_f\) which is not final optimal in one type
of final optimal labels. If we remove the last node from the drone route with the latest completion
time, the completion time of this route will be less than or equal to all other routes, otherwise the
label will not be the final optimal. For another drone route \(i\), let \(\pi^i(L_f)\) be the completion time
when removing its last node. Let \(i^* = \arg\max_{i=1,\ldots,k_d} \{\pi^i(L_f)\}\) and \(j^* = \arg\min_{j=1,\ldots,k_d} \{\pi^j(L_f)\}\). If
there exists a route \(j\) satisfying \(\pi^{i^*}(L_f) > \pi^{j^*}(L_f)\), then we move the last node in route \(i^*\) to route \(j^*\). Repeat these steps until no node needs to be moved. We denote the new label as \(L_f^{{i^*}}\). These
movement operations only advance the service start times of removed nodes, thus \(L_f^{{i^*}}\) must be a
feasible label. As both \(\pi^{i^*}(L_f^{{i^*}})\) and \(\pi^{j^*}(L_f^{{j^*}})\) are smaller than \(\pi^i(L_f)\), \(L_f^{{i^*}}\) and \(L_f\) must belong to the
same type of final optimal labels.

We then prove, all final optimal* labels can be extended from another final optimal* label by
using the method in Proposition 2. Let \(L_f\) be a final optimal* label. We continue to use the defini-
tions of \(\pi^i(L_f)\) and \(i^*\) above, and remove the last drone node in route \(i^*\) to obtain label \(L_f^{{i^*}}\). Due
to \(\pi^{i^*}(L_f^{{i^*}}) = \min_{i=1,\ldots,k_d} \{\pi^i(L_f)\}\), we can obtain \(L_f\) from \(L_f^{{i^*}}\) using the extension method in Proposition 2. We then prove \(L_f^{{i^*}}\) is a final optimal* label. If removing the last node in route \(k\) \((k \neq i^*)\), then
we have \(\pi^k(L_f^{{i^*}}) = \pi^{i^*}(L_f) \leq \pi^{i^*}(L_f^{{i^*}}) = \pi^{i^*}(L_f^{{j^*}})\). And for route \(i^*\), the relation \(\pi^{i^*}(L_f^{{i^*}}) < \pi^{i^*}(L_f^{{j^*}}) = \pi^{i^*}(L_f) \leq \pi^k(L_f) = \pi^k(L_f^{{i^*}})\) holds, so \(L_f^{{i^*}}\) is a final optimal* label. \(\Box\)

4.3.2. Backward Labeling. Let \(p(L_b)\) be a feasible backward partial path, which is denoted
by a label \(L_b = (\nu(L_b), \sigma(L_b), \pi^i(L_b), \rho^i(L_b), \omega^i(L_b), C_1(L_b), C_2(L_b), \Omega_1(L_b), \Omega_2(L_b))\), where attributes
\(\nu(L_b), \sigma(L_b), \omega^i(L_b), \Omega_1(L_b), \Omega_2(L_b)\) have the same meanings as their counterparts in the forward label \(L_f\), and the meanings of other attributes are as follows:

- \(\pi^i(L_b)\): the latest departure time of drone \(i\) \((i = 1, \ldots, \bar{k}_d)\) from \(\sigma(L_b)\) to serve the first customer
in its route \(i\); if \(i = 0\), parameter \(\pi^i(L_b)\) is the latest arrival time of the vehicle to serve \(\sigma(L_b)\);

- \(\rho^i(L_b)\): the time duration of drone route \(i\) \((i = 1, \ldots, \bar{k}_d)\); if \(i = 0\), it represents the time when
the vehicle finishes serving \(\sigma(L_b)\);

- \(C_1(L_b)\): the accumulated dual value of the partial path \(p(L_b)\);

- \(C_2(L_b)\): the time duration from \(\sigma(L_b)\) to the end node 0.

The backward labeling starts from the initial label \(L_b = (0, 0, l_0, 0, Q - q^d_1k_d, 0, 0, V, Z_d)\). Given a
label \(L_b^{{i^*}}\), a new label \(L_b\) can be created by adding a new node \(w\) to the partial path. As in the
forwarding extension, we consider two cases to generate the new label \(L_b\):
Case 1. If \( w \in \Omega_1(L'_b) \), a feasible extension must satisfy \( \kappa(L'_b) - q_w \geq 0 \) and \( \min_{i=0, \ldots, \bar{k}_d} \{ \pi^i(L'_b) \} - t_{wv(L'_b)}^d - s_w^d \geq 0 \). The following relations are used to obtain the attributes of label \( L_b \).

\[
\begin{align*}
\nu(L_b) &= w \\
\sigma(L_b) &= v \\
\pi^i(L_b) &= \begin{cases} 
\min_{j=0, \ldots, \bar{k}_d} \{ \pi^j(L'_b) \} - t_{wv(L'_b)}^d & \text{if } i = 1, \ldots, \bar{k}_d \\
\min \{ \min_{j=0, \ldots, \bar{k}_d} \{ \pi^j(L'_b) \} - t_{wv(L'_b)}^d - s_w^d, l_w \} & \text{if } i = 0 
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\rho^i(L_b) &= \begin{cases} 
0 & \text{if } i = 1, \ldots, \bar{k}_d \\
s_w^d & \text{if } i = 0 
\end{cases}
\end{align*}
\]

\[
\kappa(L_b) = \kappa(L'_b) - q_w
\]

\[
C_1(L_b) = C_1(L'_b) + \lambda_w
\]

\[
C_2(L_b) = C_2(L'_b) + \max_{i=0, \ldots, \bar{k}_d} \{ \rho^i(L'_b) \} + t_{wv(L'_b)}^d
\]

\[
\Omega_1(L_b) = \Omega_1(L'_b) - \{ w' : (w, w') \in A \text{ and } \kappa(L_b) + q_w < 0 \} - \{ w \}
\]

\[
\Omega_2(L_b) = \Omega_2(L'_b) - \{ w' : w' \in Z_d \text{ and } \kappa(L_b) + q_w < 0 \} - \{ w \}.
\]

Case 2. If \( w \in \Omega_2(L'_b) \), a feasible extension must satisfy \( \kappa(L'_b) - q_w \geq 0 \) and \( E_{\sigma(L'_b)w} = 1 \), and there exists at least one index \( i (i = 1, \ldots, \bar{k}_d) \) such that \( \min \{ \pi^i(L'_b) - t_{wv(L'_b)}^d - s_w^d, l_w \} - t_{\sigma(L'_b)w}^d - t_0 \geq 0 \). The following relations are used to obtain the attributes of label \( L_b \).

\[
\begin{align*}
\nu(L_b) &= w \\
\sigma(L_b) &= \sigma(L'_b) \\
\pi^i(L_b) &= \begin{cases} 
\pi^i(L'_b) & \text{if } i = 0 \\
\min \{ \pi^i(L'_b) - t_{wv(L'_b)}^d - s_w^d, l_w \} - t_{\sigma(L'_b)w}^d & \text{if drone } i (i = 1, \ldots, \bar{k}_d) \text{ does not visit } w \\
\rho^i(L'_b) + t_{\sigma(L'_b)w}^d + s_w^d + t_{wv(L'_b)}^d + t_0 & \text{if drone } i (i = 1, \ldots, \bar{k}_d) \text{ visits } w 
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\rho^i(L_b) &= \begin{cases} 
0 & \text{if } i = 1, \ldots, \bar{k}_d \\
\rho^i(L'_b) & \text{if } i = 0 \\
\rho^i(L'_b) + t_{\sigma(L'_b)w}^d + s_w^d + t_{wv(L'_b)}^d + t_0 & \text{if drone } i (i = 1, \ldots, \bar{k}_d) \text{ visits } w 
\end{cases}
\end{align*}
\]

\[
\kappa(L_b) = \kappa(L'_b) - q_w
\]

\[
C_1(L_b) = C_1(L'_b) + \lambda_w
\]

\[
C_2(L_b) = C_2(L'_b)
\]

\[
\Omega_1(L_b) = \Omega_1(L'_b) - \{ w' : (w, w') \in A \text{ and } \kappa(L_b) + q_w < 0 \} - \{ w \}
\]

\[
\Omega_2(L_b) = \Omega_2(L'_b) - \{ w' : w' \in Z_d \text{ and } \kappa(L_b) + q_w < 0 \} - \{ w \}.
\]

Similar to the forward labeling algorithm, we generate all the feasible labels when extending \( L'_b \). The dominance rule used in backward labeling is given in Proposition 3.
Proposition 3 \( L_1 \prec L_2 \) if the following conditions are satisfied

\[
\begin{align*}
\sigma(L_1) &= \sigma(L_2) \\
\pi^i(L_1) &\geq \pi^i(L_2) \quad i = 0, \ldots, k_d \\
\kappa(L_1) &\geq \kappa(L_2) \\
\Omega_1(L_2) &\subseteq \Omega_1(L_1) \\
\Omega_2(L_2) &\subseteq \Omega_2(L_1) \\
\rho^i(L_1) + C_2(L_1) - C_1(L_1) - \rho^i(L_2) - C_2(L_2) + C_1(L_2) &\leq 0 \quad i = 0, \ldots, k_d
\end{align*}
\]

Proof of Proposition 3: Here we only give the proof of \( \bar{C}(p_1) \leq \bar{C}(p_2) \). The proof of \( P(L_2) \subseteq P(L_1) \) can refer to the proof of Proposition 1. Let \( P(L_b) \) be the set of all feasible forward partial paths that can be connected with \( p(L_b) \) to produce at least one feasible completed route. Let \( p^f \in P(L_b) \) be one of the partial paths. For notational simplicity, we let attributes \( \pi^f(p^f) \) and \( C(p^f) \) have the same meanings as \( \pi^i(L_f) \) and \( C(L_f) \), respectively. Then we have

\[\begin{align*}
\bar{C}(p_1) - \bar{C}(p_2) &= \max \left\{ \max_{i=1, \ldots, k_d} \left\{ \pi^f(p^f) + \rho^i(L_b) \right\} , \pi^0(p^f) \right\} + C_2(L_1) - C(p^f) - C_1(L_1) \\
&\quad - \max \left\{ \max_{i=1, \ldots, k_d} \left\{ \pi^f(p^f) + \rho^i(L_b) \right\} , \pi^0(p^f) \right\} - C_2(L_2) + C(p^f) + C_1(L_2) \\
&= \max \left\{ \max_{i=1, \ldots, k_d} \left\{ \pi^f(p^f) + \rho^i(L_b) + C_2(L_1) - C_1(L_1) \right\} , \pi^0(p^f) + C_2(L_1) - C_1(L_1) \right\} \\
&\quad - \max \left\{ \max_{i=1, \ldots, k_d} \left\{ \pi^f(p^f) + \rho^i(L_b) + C_2(L_1) - C_1(L_1) \right\} , \pi^0(p^f) + C_2(L_2) - C_1(L_2) \right\} \\
&\leq \max \left\{ \max_{i=1, \ldots, k_d} \left\{ \pi^f(p^f) + \rho^i(L_1) + C_2(L_1) - C_1(L_1) \right\} \right\} \\
&\quad - \max \left\{ \max_{i=1, \ldots, k_d} \left\{ \pi^f(p^f) + \rho^i(L_2) + C_2(L_2) - C_1(L_2) \right\} \right\} \\
&\leq \max \left\{ \max_{i=0, \ldots, k_d} \left\{ \rho^i(L_1) + C_2(L_1) - C_1(L_1) - \rho^i(L_2) - C_2(L_2) + C_1(L_2) \right\} \right\} \\
&= \max \left\{ \max_{i=0, \ldots, k_d} \left\{ \rho^i(L_1) + C_2(L_1) - C_1(L_1) - \rho^i(L_2) - C_2(L_2) + C_1(L_2) \right\} \right\} \\
&\leq 0.
\end{align*}\]

In the backward labeling algorithm, the feasibility of deadline constraints and time duration of a drone route \( i \) are evaluated by two series of attributes \( \pi^f(L_b) \) and \( \rho^f(L_b) \), which may not have the same order in their series. Thus, we cannot use Corollary 1 to simplify the dominance rule. Proposition 2 is also not applicable for the same reason.
4.3.3. Label Joining. When $\bar{k}_d = 0$, a forward label and a backward label can be extended only when $\pi^0(L_f) - \sigma^0(L_f) < l_0/2$ and $\pi^0(L_f) > l_0/2$, respectively. When $\bar{k}_d \geq 1$, a forward label and a backward label can be extended only when $\min_{i=1,\ldots,\bar{k}_d} \{\pi^i(L_f)\} < l_0/2$ and $\min_{i=1,\ldots,\bar{k}_d} \{\pi^i(L_b)\} > l_0/2$, respectively. A forward label $L_f$ and a backward label $L_b$ can be merged into a feasible completed route if the following conditions are satisfied:

\begin{align}
\sigma(L_f) &= \sigma(L_b) \\
\pi^i(L_f) &\leq \pi^i(L_b) \quad i = 0, \ldots, \bar{k}_d \\
\kappa(L_f) + \kappa(L_b) + q^i(L_f) + q^i(L_b) &\geq Q \\
S(L_f) \cap S(L_b) &= \emptyset
\end{align}

where $S(L_f)$ and $S(L_b)$ are the sets of nodes visited along the partial paths $p(L_f)$ and $p(L_b)$, respectively. Condition (56) ensures that the two partial paths end at the same vehicle node. Conditions (57)–(59) guarantee the time relation between the drone routes, vehicle capacity constraints, and route elementarity, respectively.

For the resulting completed route, its time duration is

$$c_r = C_2(L_b) + \max \left\{ \max_{i=1,\ldots,\bar{k}_d} \{\pi^i(L_f) + \rho^i(L_b)\}, \pi^0(L_f) \right\},$$

and its reduced cost is

$$\bar{c}_r = c_r - C(L_f) - C(L_b) - \lambda^v_0 - \bar{k}_d \lambda^d.$$

When joining two labels, drone route $i$ in label $L_f$ can only be connected with route $j$ in label $L_b$ when the relation $\pi^i(L_f) \leq \pi^i(L_b)$ holds. To make $\bar{c}_r$ as small as possible, route $i$ with a larger value of $\pi^i(L_f)$ should be connected with route $j$ with a smaller value of $\rho^i(L_b)$. To find the best pair of $(i, j)$, i.e., the resulting completed route has the shortest time duration, we can use an enumeration method with a time complexity of $O(n!)$. To do it more efficiently, we propose a pairing algorithm with a time complexity of $O(n^2)$, which is described in Algorithm 2.

**Corollary 2** Algorithm 2 can always obtain the best pairs of partial drone routes.

**Proof of Corollary 2**: We assume $\pi^i(L_f)$ and $\rho^i(L_b)$ are sorted in non-ascending and non-descending orders, respectively. $\pi^i(L_b)$ are sorted in non-descending order concerning the value of $\rho^i(L_b)$. We denote $\theta(i)$ as the set of $j$ satisfying $\pi^i(L_f) \leq \pi^i(L_b)$, and sort the set in non-descending order concerning the value of $\rho^i(L_b)$. As $\pi^1(L_f) \geq \pi^2(L_f) \geq \ldots \geq \pi^{\bar{k}_d}(L_f)$, the relation $\theta(1) \subseteq \theta(2) \subseteq \ldots \subseteq \theta(\bar{k}_d)$ holds.

Let $(i^*, j^*)$ be one of the pairs found by Algorithm 2, which has the drone route with the longest time duration. We first prove that $j^*$ must be the $i^{th}$ route in set $\theta(i^*)$. For all the pairs produced
Algorithm 2 The pseudo code of the pairing algorithm.

1: Input: forward label $L_f$ and backward label $L_b$.
2: Define sets $\mathcal{F}$ and $\mathcal{B}$: partial drone routes associated with $L_f$ and $L_b$, respectively; $P$: set of pairs of partial drone routes.
3: Sort $\mathcal{F}$ in non-ascending order concerning the value of $\pi^i(L_f)$; Sort $\mathcal{B}$ in non-descending order concerning the value of $\rho^i(L_b)$.
4: for all route $f \in \mathcal{F}$ do
5:  for all route $b \in \mathcal{B}$ do
6:      if $\pi(f) \leq \pi(b)$ then
7:          $\text{if} \pi(f) \leq \pi(b)$ \text{then}
8:          \textbf{end if}
9:      \textbf{end for}
10: \textbf{end for}
11: Output: set $P$.

by Algorithm 2, a forward route $i$ must be paired with one route before the $(i + 1)^{th}$ route in set $\theta(i)$, because forward routes $1, 2, \ldots, i - 1$ can only be paired with at most the first $i - 1$ routes in set $\theta(i)$. We denote $i_k$ ($k \geq 1$) as the forward route which is paired with the $i_k^{th}$ route in set $\theta(i_k)$. There must exist such a $i_k$, because the forward route $i = 1$ must be paired with the first route in $\theta(1)$. Then, for a forward route $i$ ($i_k < i < i_k + 1$), its drone route duration must be less than that of route $i_k$. Thus, the route with the longest time duration must be $i_k$ ($k \geq 1$).

Next, we prove that Algorithm 2 can find the best pairs by contradiction. We assume there exist better pairing results. In these pairs, $i^*$ should be paired with a backward drone route ranking before $i^*$ in $\theta(i^*)$, i.e., $j^*$. Then the resulting time duration can be smaller than $\pi^*(L_f) + \rho^*(L_b)$. For a forward route $i$ ($i < i^*$), as $\theta(i) \subseteq \theta(i^*)$, $i$ can be paired with at most the first $i^* - 1$ routes in $\theta(i^*)$, otherwise the route duration will exceed $\pi^*(L_f) + \rho^*(L_b)$. Therefore, $i^* - 1$ forward routes should be paired with $i^* - 2$ backward routes, which is impossible. 

4.3.4. Heuristic Column Generation by Tabu Search. The labeling algorithm is often time-consuming. To accelerate the CG procedure, we can apply some heuristic methods instead of the exact labeling algorithm to find routes with negative reduced costs. This section introduces a TS algorithm for this purpose (Desaulniers et al. 2008).

When solving the pricing problem, the TS algorithm is first applied. If a fixed number of routes with negative reduced costs are found, the labeling algorithm will be skipped. The TS starts from existing columns with reduced costs being 0. We use three operators to improve the solution: \textit{insertion}, \textit{removal}, and \textit{shift}. The insertion operator attempts to insert an unserved customer into a route. The removal operator removes a drone node or vehicle node not connected with any
drone node. The insertion and removal operators are commonly used in the literature, while the shift operator is specially designed for the ESPPDRC. For all drone routes from a vehicle node, we consider shifting the last node in the drone route with the latest completion time to the drone route with the earliest completion time. If the time duration at the vehicle node can be reduced, the shift operation is accepted. Figure 3 shows an example of the shift operator, where two drones start from the vehicle node to serve four customers. The values around the customers are the completion times of the corresponding trips. If we shift customer 4 in the route of drone 1 to the route of drone 2, the makespan can be decreased from 20 to 16. Thus, this shift operation is accepted. The tabu list forbids the arcs for insertion and removal operators. Shift operator will not be forbidden but only be applied when the duration at the vehicle node can be reduced. More details about the TS algorithm can refer to Glover (1989) and Glover and Laguna (1998).

![Image of shift operator example](image)

**Figure 3** An example of the shift operator used in the TS algorithm.

### 4.4. Branching Strategy

We use a best-first strategy to explore the B&B tree, i.e., when selecting a tree node to explore, we first consider the one whose father node has the smallest lower bound. We use the following hierarchical strategies to branch on fractional variables:

1. **The number of used vehicles.** Let \( \bar{H}_v = \sum_{r \in R} \bar{\mu}_r \) represent the number of vehicles used in the optimal solution of the LRP. If \( \bar{H}_v \) is fractional, we create two child nodes with constraints \( \sum_{r \in R} \bar{\mu}_r \leq \lceil \bar{H}_v \rceil \) and \( \sum_{r \in R} \bar{\mu}_r > \lceil \bar{H}_v \rceil \), respectively.

2. **The number of used drones.** Let \( \bar{H}_d = \sum_{r \in R} \bar{d}_r \bar{\mu}_r \) represent the number of drones used in the optimal solution of the LRP. If \( \bar{H}_d \) is fractional, we create two child nodes with constraints \( \sum_{r \in R} \bar{d}_r \bar{\mu}_r \leq \lceil \bar{H}_d \rceil \) and \( \sum_{r \in R} \bar{d}_r \bar{\mu}_r > \lceil \bar{H}_d \rceil \), respectively.

3. **The flow between two vehicle nodes.** Let \( \bar{x}_{ij} = \sum_{r \in R} \gamma_{ij} \bar{\mu}_r \) be the total flow between two vehicle nodes \( i \) and \( j \), where \( \gamma_{ij} \) represents whether route \( r \) traverses arc \( (i, j) \). We branch on arc \( (i^*, j^*) \)
whose value of $\tilde{x}_{ijr}$ is the closest to 0.5. Two child nodes are created by adding constraints
\[ \sum_{r \in R} \gamma_{ijr} \mu_r \leq \lfloor \tilde{x}_{ijr} \rfloor \quad \text{and} \quad \sum_{r \in R} \gamma_{ijr} \mu_r \geq \lceil \tilde{x}_{ijr} \rceil. \]

4. **The flow between a vehicle node and a drone node.** Let $\tilde{u}_{ij} = \sum_{r \in R} \xi_{ijr} \tilde{\mu}_r$ be the total flow between vehicle node $i$ and drone node $j$, where parameter $\xi_{ijr}$ represents whether node $j$ is served by a drone starting from node $i$ in route $r$. We branch on arc $(i^*, j^*)$ whose value of $\tilde{u}_{i^*j^*}$ is the closest to 0.5. Two child nodes are created by adding constraints \[ \sum_{r \in R} \xi_{i^*j^*r} \mu_r \leq \lfloor \tilde{u}_{i^*j^*} \rfloor \quad \text{and} \quad \sum_{r \in R} \xi_{i^*j^*r} \mu_r \geq \lceil \tilde{u}_{i^*j^*} \rceil. \]

5. **Computational Experiments**

This section presents the instance sets and analyzes the performance of the B&P algorithm. Our algorithm was coded in Java programming language using ILOG CPLEX 12.6.3 as the solver. The experiments were conducted on a machine equipped with a 3.3GHz Intel(R) Xeon(R) Platinum 8369HB CPU and 32G of memory under the Windows 10 operating system. The time limit on each run of the B&P algorithm was set to 3600 and 10800 seconds for small- and medium-size instances, respectively.

5.1. **Instances**

We use two instance sets to evaluate our algorithm. The first benchmark instance set is introduced by Chen et al. (2021a), which is available at https://data.mendeley.com/v1/datasets/kxfcwkwdb9/draft?a=edb5ce79-b4c7-4121-93ca-317e82328b1c. The authors created these instances by randomly choosing Postcodes from Cardiff, the United Kingdom for customer locations, and then generating demand and time window for each customer. Their instances contain seven sets, each with 15, 25, 50, ..., and 200 customers. For a fixed number of customers, 20 instances are generated. We consider the instances with 15 and 25 customers as small- and medium-size instances, respectively. We further generate instances with 10 customers by removing the last five customers of the 15-customer instances. Considering the capacity of our drones, we set $f_i^d = 0$ if $q_i \geq 10$, i.e., if the demand of a customer is equal to or larger than 10, it cannot be visited by a drone. To simulate other cases where drone delivery is infeasible, we randomly select $\lfloor 0.2n \rfloor$ customers from the rest customers and set them as drone-forbidden customers. We name instances in this set Cardiff$\_n$0, where $n$ denotes the number of customers and 0 is the order of the instance. The second instance set is based on the well-known Solomon (1987) instances. We generate medium-size instances by adopting the data of the first 35 customers of their type C instances. We set $s_i^d = 0.5s_i^c$, $i \in Z$ and name instances in this set c10X$\_n$, where c10X is the original instance name in Solomon (1987) and $n$ is the number of customers. For 10- and 15-customer instances, we set $Q = 170$. For 25- and 35-customer instances, we set $Q = 200$ and 250, respectively. For small- and medium-size instances, we set $k_0 = 2$ and 3, respectively.
We let $g = 9.81 \text{ N/kg}$ and $\rho = 1.204 \text{ kg/m}^3$ as in Dorling et al. (2016) and Cheng et al. (2020). Drone related parameters are collected from the Ark Octocopter Drone, which is designed by SF Technology (https://www.sf-tech.com.cn/) to perform short- to mid-range deliveries. The values of parameters $\zeta$ and $B_c$ are not given in the website. Thus, we estimate $\zeta = 0.2 \text{ m}^2$ based on the picture of the drone as shown in Figure 4. We set $B_c = 0.6 \text{ kWh}$ for the first instance set, which indicates that the delivery radius is 1100 meters on average. For the second instance set, we set $B_c = 0.06 \text{ kWh}$, because the distance between customers is much smaller than that in the first instance set. Other parameters are set as follows: $q^1_d = 50 \text{ kg}, q^2_d = 37 \text{ kg}, \text{vel}^d = 15 \text{ m/s}, \text{vel}^p = 12 \text{ m/s}$, and $h = 8$.

5.2. Algorithm Performance

In this section, we first use small-size instances to evaluate the performance of the B&P algorithm by comparing it with the MILP model solved by CPLEX. We then explore the performance of the B&P for medium-size instances. In the following tables, columns $UB$ and $LB$ are the best upper and lower bounds produced by B&P or CPLEX. $Gap$ is the percentage difference between the best upper and lower bounds. $#Node$ is the number of nodes explored in the B&B tree. Column $Time$ reports the time in seconds consumed to solve one instance.

5.2.1. Performance Comparison between B&P and CPLEX. Table 2 reports the results of small-size instances, where column $\Delta_{ub}$ is the percentage difference between the best upper bounds produced by the two methods. Specifically, $\Delta_{ub} = (UB_{cplex} - UB_{BP}) / UB_{cplex} \times 100$, where $UB_{cplex}$ and $UB_{BP}$ are produced by CPLEX and B&P, respectively. Similarly, $\Delta_{lb} = (LB_{cplex} - LB_{BP}) / LB_{cplex} \times 100$. Note that a positive value of $\Delta_{ub}$ suggests that B&P has provided a better feasible solution, and a negative value of $\Delta_{lb}$ indicates that B&P has produced a tighter lower bound. Boldface letters are used to indicate the shorter computing times.

Table 2 shows that both methods can solve all the 10-customer instances to optimality within the time limit. However, the average computing time of B&P is 2.6 seconds, which is much shorter than that of CPLEX, i.e., 121.2 seconds. Notably, CPLEX has explored a large number of B&B nodes, i.e., 167950.8 on average, whereas B&P only explores 5.6 nodes on average. For 15-customer
Table 2 Performance comparison between B&P and CPLEX on small-size instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>B&amp;P UB</th>
<th>B&amp;P LB</th>
<th>B&amp;P Gap</th>
<th>CPLEX UB</th>
<th>CPLEX LB</th>
<th>CPLEX Gap</th>
<th>#Node</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardiff.10.01</td>
<td>1223.6</td>
<td>1223.6</td>
<td>0.00</td>
<td>1223.6</td>
<td>1223.6</td>
<td>0.00</td>
<td>2</td>
<td>2.2</td>
</tr>
<tr>
<td>Cardiff.10.02</td>
<td>1905.1</td>
<td>1905.1</td>
<td>0.00</td>
<td>1905.1</td>
<td>1905.1</td>
<td>0.00</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Cardiff.10.03</td>
<td>1412.7</td>
<td>1412.7</td>
<td>0.00</td>
<td>1412.7</td>
<td>1412.7</td>
<td>0.00</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>Cardiff.10.04</td>
<td>1652.7</td>
<td>1652.7</td>
<td>33.21</td>
<td>1652.7</td>
<td>1652.7</td>
<td>0.00</td>
<td>33</td>
<td>21.5</td>
</tr>
<tr>
<td>Cardiff.10.05</td>
<td>1682.9</td>
<td>1682.9</td>
<td>0.00</td>
<td>1682.9</td>
<td>1682.9</td>
<td>0.00</td>
<td>1</td>
<td>0.4</td>
</tr>
<tr>
<td>Cardiff.10.06</td>
<td>1221.8</td>
<td>1221.8</td>
<td>0.00</td>
<td>1221.8</td>
<td>1221.8</td>
<td>0.00</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Cardiff.10.07</td>
<td>1931.8</td>
<td>1931.8</td>
<td>0.00</td>
<td>1931.8</td>
<td>1931.8</td>
<td>0.00</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Cardiff.10.08</td>
<td>1578.1</td>
<td>1578.1</td>
<td>0.00</td>
<td>1578.1</td>
<td>1578.1</td>
<td>0.00</td>
<td>37</td>
<td>7.5</td>
</tr>
<tr>
<td>Cardiff.10.09</td>
<td>2552.0</td>
<td>2552.0</td>
<td>0.00</td>
<td>2552.0</td>
<td>2552.0</td>
<td>0.00</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>Cardiff.10.10</td>
<td>2045.2</td>
<td>2045.2</td>
<td>0.00</td>
<td>2045.2</td>
<td>2045.2</td>
<td>0.00</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>Cardiff.10.11</td>
<td>2095.3</td>
<td>2095.3</td>
<td>0.00</td>
<td>2095.3</td>
<td>2095.3</td>
<td>0.00</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Cardiff.10.12</td>
<td>1737.3</td>
<td>1737.3</td>
<td>0.00</td>
<td>1737.3</td>
<td>1737.3</td>
<td>0.00</td>
<td>24</td>
<td>16.0</td>
</tr>
<tr>
<td>Cardiff.10.13</td>
<td>1579.2</td>
<td>1579.2</td>
<td>0.00</td>
<td>1579.2</td>
<td>1579.2</td>
<td>0.00</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Cardiff.10.14</td>
<td>1926.6</td>
<td>1926.6</td>
<td>0.00</td>
<td>1926.6</td>
<td>1926.6</td>
<td>0.00</td>
<td>1</td>
<td>0.8</td>
</tr>
<tr>
<td>Cardiff.10.15</td>
<td>1814.3</td>
<td>1814.3</td>
<td>0.00</td>
<td>1814.3</td>
<td>1814.3</td>
<td>0.00</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>Cardiff.10.16</td>
<td>1150.5</td>
<td>1150.5</td>
<td>0.00</td>
<td>1150.5</td>
<td>1150.5</td>
<td>0.00</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>Cardiff.10.17</td>
<td>1733.8</td>
<td>1733.8</td>
<td>0.00</td>
<td>1733.8</td>
<td>1733.8</td>
<td>0.00</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Cardiff.10.18</td>
<td>1960.9</td>
<td>1960.9</td>
<td>0.00</td>
<td>1960.9</td>
<td>1960.9</td>
<td>0.00</td>
<td>1</td>
<td>0.2</td>
</tr>
<tr>
<td>Cardiff.10.19</td>
<td>1533.7</td>
<td>1533.7</td>
<td>0.00</td>
<td>1533.7</td>
<td>1533.7</td>
<td>0.00</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>Cardiff.10.20</td>
<td>1366.4</td>
<td>1366.4</td>
<td>0.00</td>
<td>1366.4</td>
<td>1366.4</td>
<td>0.00</td>
<td>1</td>
<td>0.6</td>
</tr>
<tr>
<td>Average</td>
<td>1705.2</td>
<td>1705.2</td>
<td>0.00</td>
<td>1705.2</td>
<td>1705.2</td>
<td>0.00</td>
<td>5.6</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Instances: B&P can solve all the instances to optimality, consuming 9.2 seconds on average. However, no instance can be optimally solved by CPLEX, and the average optimality gap is 35.58%.

When comparing the final upper bounds provided by the two methods, we find that CPLEX has found the optimal solution for 10 instances; however, it fails to prove the optimality of solutions due to the poor quality of the lower bounds, as indicated in the last column.
5.2.2. Performance of B&P for Medium-size Instances. We consider two cases to evaluate the performance of B&P for medium-size instances. The first case is the VRPD defined in this study, and the second case is the VRPD with one drone on each truck. Results are reported in Table 3. We note that some authors have studied the second-case problem, e.g., Schermer et al. (2018) and Sacramento et al. (2019), but heuristic algorithms are often applied.

Table 3 shows that for the VRPD defined in this study, B&P can solve 17 out of 20 instances to optimality for 25-customer instances. For the other three instances, i.e., Cardiff.25.04, Cardiff.25.12, and Cardiff.25.14, B&P can provide good solutions within the time limit—the optimality gaps are 2.17%, 2.75%, and 5.25%, respectively. For 35-customer instances, B&P can solve 4 out of 9 instances to optimality. For instance c102.35, although it is not optimally solved, its optimality gap is quite small—only 0.24%. For the rest four instances, the gap is relatively large. However, we consider the obtained upper and lower bounds could be useful for evaluating heuristic methods. For the VRPD with one drone on each truck, B&P can solve 19 out of 20 (8 out of 9) instances to optimality for 25- (35-) customer instances. For instances Cardiff.25.14 and c104.35, the optimality gaps are small—1.97% and 0.57%, respectively. Thus, we conclude that our B&P can solve most medium-size instances efficiently, especially for the VRPD with one drone on each truck.

5.3. Value of Drone Allocation

One of the main characteristics of our VRPD is that we consider the allocation decisions of drones. In contrast, the number of drones carried by each vehicle is set as a parameter in the literature. This section quantifies the benefits (i.e., the savings of time duration) from the allocation decisions. For the case of non-allocation decisions, we fix the number of paired drones on each vehicle to $A_0 = k_d / k_v$. Experiments are performed on instances with 10, 15, and 25 customers that are optimally solved, and results are reported in Table 4. Gaps are computed as $(D_n - D_o) / D_n \times 100$, where $D_n$ and $D_o$ are the durations of all routes under the non-allocation and allocation decisions, respectively. The last column is Table 4 reports the maximal gap among all the instances in that set.

Table 4 shows that the allocation decisions can help save the total duration, from 0.26% for 10-customer instances to 1.01% for 25-customer instances on average. The maximal saving can reach 1.80%, 6.59%, and 6.70% for some 10-, 15-, and 25-customer instances, respectively. In addition, we notice that the values of $k_0$ and $A_0$ are very close; that is, compared to the case of non-allocation decisions, the optimization space left for the allocation decisions is relatively small. We try to reduce the number of available drones in the system such that $A_0 = 1$ (i.e., one drone is allocated to each truck) for the 25-customer instances. Results show that the average time-saving increases to 3.45%, and the maximal gap can reach 11.48%. This phenomenon suggests that our allocation decisions are particularly beneficial for a delivery system with limited drones.
Table 3  Performance of B&P on medium-size instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>VRPD defined in this study</th>
<th>VRPD with one drone on each truck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UB</td>
<td>LB</td>
</tr>
<tr>
<td>Cardiff_25.01</td>
<td>2233.9</td>
<td>2233.9</td>
</tr>
<tr>
<td>Cardiff_25.02</td>
<td>1980.0</td>
<td>1980.0</td>
</tr>
<tr>
<td>Cardiff_25.03</td>
<td>2389.7</td>
<td>2389.7</td>
</tr>
<tr>
<td>Cardiff_25.04</td>
<td>2814.7</td>
<td>2753.6</td>
</tr>
<tr>
<td>Cardiff_25.05</td>
<td>2063.0</td>
<td>2063.0</td>
</tr>
<tr>
<td>Cardiff_25.06</td>
<td>2075.5</td>
<td>2075.5</td>
</tr>
<tr>
<td>Cardiff_25.07</td>
<td>2112.6</td>
<td>2112.6</td>
</tr>
<tr>
<td>Cardiff_25.08</td>
<td>2605.5</td>
<td>2605.5</td>
</tr>
<tr>
<td>Cardiff_25.09</td>
<td>2055.2</td>
<td>2055.2</td>
</tr>
<tr>
<td>Cardiff_25.10</td>
<td>2220.7</td>
<td>2220.7</td>
</tr>
<tr>
<td>Cardiff_25.11</td>
<td>2051.1</td>
<td>2051.1</td>
</tr>
<tr>
<td>Cardiff_25.12</td>
<td>4783.1</td>
<td>4651.4</td>
</tr>
<tr>
<td>Cardiff_25.13</td>
<td>2297.5</td>
<td>2297.5</td>
</tr>
<tr>
<td>Cardiff_25.14</td>
<td>2561.4</td>
<td>2426.9</td>
</tr>
<tr>
<td>Cardiff_25.15</td>
<td>2871.6</td>
<td>2871.6</td>
</tr>
<tr>
<td>Cardiff_25.16</td>
<td>2503.3</td>
<td>2503.3</td>
</tr>
<tr>
<td>Cardiff_25.17</td>
<td>2275.8</td>
<td>2275.8</td>
</tr>
<tr>
<td>Cardiff_25.18</td>
<td>2167.9</td>
<td>2167.9</td>
</tr>
<tr>
<td>Cardiff_25.19</td>
<td>2257.8</td>
<td>2257.8</td>
</tr>
<tr>
<td>Cardiff_25.20</td>
<td>2895.6</td>
<td>2895.6</td>
</tr>
</tbody>
</table>

Table 4  Duration saving from drone allocation decisions.

<table>
<thead>
<tr>
<th>Instance</th>
<th>$k_0$</th>
<th>$A_0$</th>
<th>Non-allocation</th>
<th>Allocation</th>
<th>Gap (%)</th>
<th>MaxGap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardiff_10</td>
<td>2</td>
<td>1</td>
<td>1709.61</td>
<td>1705.19</td>
<td>0.26</td>
<td>1.80</td>
</tr>
<tr>
<td>Cardiff_15</td>
<td>2</td>
<td>1</td>
<td>2028.96</td>
<td>2012.97</td>
<td>0.79</td>
<td>6.59</td>
</tr>
<tr>
<td>Cardiff_25</td>
<td>3</td>
<td>2</td>
<td>2320.54</td>
<td>2297.45</td>
<td>1.01</td>
<td>6.70</td>
</tr>
</tbody>
</table>

5.4. Sensitivity Analyses

Since our VRPD is a new variant of the truck-drone routing problem, we conduct sensitivity analyses on key parameters, including battery energy capacity $B_c$, the maximum number of drones that a vehicle can carry $k_0$, and drone speed $vel^d$, to investigate the characteristics of this variant. Experiments are performed on 10- and 15-customer instances.

Battery Energy Capacity. In drone-aided routing problems, the battery energy capacity $B_c$ influences the number of drone-served customers, and thus the time duration of all routes. We analyze the influence of this parameter by changing its value from 0.6 to 1.2 with a step size of 0.2. Fig-
Figure 5 plots the average results, where the duration saving is calculated with respect to the result under $B_c = 0.6$. We can observe that batteries of a higher energy density, i.e., an enhanced battery technique, can increase the number of drone-served customers and thus reduce the duration. In particular, when $B_c$ increases from 0.6 to 1.2, the time duration can be reduced by around 3.4% and 0.95% for 10- and 15-customer instances, respectively.

**Maximum Allowable Number of Drones on a Vehicle.** When $k_0 = 0$, the VRPD reduces to a VRP with deadlines. Now we analyze the benefits of using a truck-drone system with different values of $k_0$. As drones and their support equipment also occupy vehicle capacity, we consider two different values for parameter $Q$, i.e., $Q = 170$ and $Q = 220$. The average results are reported in Figure 6, where the duration saving is calculated with respect to the case of $k_0 = 0$. Figure 6 shows that the truck-drone system can help reduce the total duration as expected, compared to the truck-only system. We notice that when $Q = 170$, i.e., the vehicle capacity is relatively small, the comparison results keep stable when $k_0$ increases from 2 to 3. The main reason is that if more drones are carried by a vehicle, the residual capacity for parcels will become smaller; thus, even though a vehicle is allowed to carry three drones, it might be more beneficial to carry only two drones to save space for serving more customers. When $Q$ is increased to 220, more time can be saved when $k_0$ varies from 2 to 3, particularly for the 15-customer instances.

**Drone Speed.** Now, we study the impact of drone speed $vel^d$. The average results are plotted in Figure 7, where the duration saving is computed with respect to the case of $vel^d = 12$. When $vel^d = 12$, i.e., drone speed and vehicle speed are identical, only a minimal number of customers are served by drones. However, when the drone speed increases, more customers are assigned to get service from drones, resulting in more duration savings. In particular, when $vel^d = 18$, the time duration can be reduced by 4.4% and 4.0% for 10- and 15-customer instances, respectively.
Figure 6  The impact of the maximum allowable number of drones on a vehicle.

Figure 7  The impact of drone speed.
To summarize, multiple strategies can be used to further improve the operational efficiency of a truck-drone system, such as adopting batteries of a higher energy density, increasing the maximum allowable number of drones on a vehicle when the vehicle capacity is relatively large, and developing powerful drones with a higher flying speed.

6. Conclusions

This paper introduces a new variant of the VRPD, which extends the classic truck-drone delivery problem by considering several practical constraints, including customers’ deadlines and drones’ energy capacity. Unlike prior studies, we treat the number of drones carried by each vehicle as a decision variable instead of a given parameter. An MILP formulation is first built to model the problem, which is solvable by CPLEX. To tackle VRPD instances more efficiently, we next construct a set partitioning model solved by a B&P algorithm. A bidirectional labeling algorithm is designed to solve the pricing problem of the CG procedure. Extensive numerical tests based on two types of benchmark instances are conducted to evaluate the performance of the B&P algorithm and explore the impacts of critical variables and parameters. Results show that our B&P algorithm can solve most instances within 25 customers to optimality in a short time frame and some instances with 35 customers to optimality within the three-hour time limit. Moreover, the drone allocation decisions can help save the time duration of all routes, particularly when the number of available drones is limited in the delivery system. Results also show that multiple strategies can be applied to further improve the delivery efficiency of a truck-drone system.

References


