An Exact Algorithm for the Two-echelon Vehicle Routing Problem with Drones

Hang Zhou, Hu Qin
School of Management, Huazhong University of Science and Technology, Wuhan 430074, China

Chun Cheng*
Institute of Supply Chain Analytics, Dongbei University of Finance and Economics, Dalian 116025, China

Louis-Martin Rousseau
Polytechnique Montréal and CIRRELT, Montreal H3C 3A7, Canada

This paper studies a new variant of the vehicle routing problem with drones, i.e., the two-echelon vehicle routing problem with drones, where multiple vehicles and drones work collaboratively to serve customers. Drones can perform multiple back-and-forth trips when their paired vehicle stops at a customer node, forming a two-echelon network. 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 first formulate this problem as a mixed-integer linear programming model, which is solvable by off-the-shelf commercial solvers. To tackle instances more efficiently, we next construct a set-partitioning model. To solve it, an exact branch-and-price algorithm is proposed, where a bidirectional labeling algorithm is used to solve the pricing problem. To speed up the algorithm, 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 3.44% 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; two-echelon vehicle routing 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

*Corresponding author. Email: chun.cheng@polymtl.ca

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/or retrieve drones. Several companies have already adopted this innovative concept, like Mercedes-Benz (Unmanned Airspace 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's 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). Most truck-drone delivery systems require that a truck moves to a new location to retrieve a drone. Namely, when a drone is out for delivery, the truck must continue its trip. Although this tandem delivery mode is appealing in saving service completion time, it requires synchronization between a truck and a drone, which is not easy to implement in practical applications. Because the travels of trucks are heavily affected by traffic conditions, which may cause late arrivals. Moreover, this delivery mode disables the driver from continuously monitoring the drone, causing a potential threat to residents in urban areas. Other truck-drone delivery systems impose that trucks must wait at customer locations to retrieve drones (Kang and Lee 2021, Dukkanci et al. 2021, Moshref-Javadi et al. 2020). This mode

is named the two-echelon vehicle routing problem with mobile satellites in Li et al. (2020) because it assumes the characteristics of the two-echelon vehicle routing problem (2E-VRP). Compared with the tandem system, this two-echelon mode allows the driver to monitor the flying drones to ensure safety. Sometimes this is necessary because the current technological and regulatory environment restricts the fully dynamic and synchronized control of multi-modal drone-based delivery models. For instance, the U.S. Federal Aviation Administration requires that drones fly within an operator's sight range. Considering the technologies of drones are not sufficiently mature, the two-echelon delivery pattern may be more suitable for the truck-drone systems in urban logistics. However, to the best of our knowledge, only limited studies consider optimizing related problems.

Our Contributions. The existing research on the VRPD or the 2E-VRP often sets the number of drones taken by a vehicle or the number of trucks in the satellites to a fixed value, resulting in less flexibility for allocating drones/vehicles and thus planning 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 research gaps in drone delivery, this paper introduces a new variant of the VRPD from the perspective of the 2E-VRP, which is an intersection of these two types of problems. We named it the two-echelon vehicle routing problem with drones (2E-VRP-D). We consider our work makes the following contributions to the literature:

(1) We introduce a new variant of the VRPD, i.e., the 2E-VRP-D. 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 branch-and-price (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 benchmark instances are conducted to evaluate the performance of our B&P algorithm and the values of drone delivery and 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 defines the problem and constructs mathematical models. Section 4 presents the B&P algorithm. Section 5 describes the instance sets and provides the numerical results. Section 6 concludes this paper and suggests future research directions.

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. Cavani et al. (2021) consider the traveling salesman problem (TSP) with multiple drones. A branch-and-cut (B&C) algorithm is proposed, which can solve instances with 24 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 most truck-drone systems 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 Tiniç (2020) propose a VRP with flexible drones. The authors assume that drones can be launched and retrieved from 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. Zhen et al. (2022) extend the work of Roberti and Ruthmair (2021) from a single truck to multiple trucks, where each truck is equipped with one drone. The authors propose a branch-price-and-cut algorithm, where two types of valid inequalities, i.e., the rounded capacity inequalities and the subset-row inequalities, are added to the restricted master problem to strengthen the lower bound provided by the column generation (CG) algorithm. However, due to the complexity of the problem, their algorithm can only solve instances with up to 14 customers and 3 trucks to optimality. Di Puglia Pugliese et al. (2021a) consider the uncertain energy consumption in the VRPD with time windows, which is influenced by the weather information. A robust optimization technique is used, where an over-conservative formulation and a budgeted polytope are applied to model the uncertainty. A Benders decomposition approach is developed, which can solve 15-customer instances to optimality. Some authors also design heuristic or metaheuristic algorithms. For example, Di Puglia Pugliese et al. (2021b) consider partial time window constraints in the VRPD, which means that the time windows will only be considered when customers are served by trucks, because drones can deliver parcels on a customer's balcony without the presence of that customer. They minimize the transportation costs of both trucks and drones. A two-phase heuristic is proposed. Kuo et al. (2022) consider all customers' time windows, where each truck is equipped with a single drone. They minimize the total travel costs and design a variable neighborhood search algorithm. Li et al. (2022) also consider the time window constraints, where each truck carries a predetermined number of drones. The authors allow arc synchronization, i.e., drones can be launched when the truck is moving on the roads. An adaptive large neighborhood search (ALNS) heuristic is developed.

Only a few studies stand between the intersection of the TSP-D and the 2E-VRP. Moshref-Javadi et al. (2020) propose the multi-trip traveling repairman problem with drones, where only one truck is involved in the delivery system, and the truck can move to the next customer only after all drones are back. The authors develop a hybrid algorithm based on simulated annealing and TS. Salama and Srinivas (2020) consider a single truck with multiple drones. They partition delivery locations into small clusters and then decide on a focal point for each cluster. Subsequently, the truck is routed to visit all focal points such that customers in each cluster are visited by the truck or by a drone. They optimize the number of drones carried by the truck. Since only one truck is used, no allocation decision is involved with their problem. Considering two different

policies for choosing focal points, the authors propose mathematical models to jointly optimize all the decisions involved. A machine learning-based heuristic is designed as a warm-start procedure to accelerate the resolution of the mathematical model. 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. The authors develop a B&C algorithm based on a logic-based Benders decomposition approach, which can solve 50-customer instances to optimality. Vu et al. (2022) explore the problem with one truck and multiple identical drones. The truck does not serve customers directly, i.e., it is used as a mobile depot to assist drones in making deliveries. When drones are set out, the truck must stay at the intermediate depot until all drones return. The authors introduce a metaheuristic based on a greedy randomized adaptive search procedure.

Some authors consider the cases of multi-trucks within a two-echelon setting. Dukkanci et al. (2021) propose an energy minimizing and range constrained drone delivery problem, where trucks carry drones to the parking locations, acting as dispatch points for drones. Drones can be launched multiple times from a parking lot. The authors assume that each truck trip only visits one parking lot and determine drones' travel speeds to minimize the total variable cost, comprising the operational cost of trucks and the energy consumption cost of drones. A nonlinear model is proposed and reformulated as a second-order cone programming (SOCP) formulation. Kitjacharoenchai et al. (2020) study the case where each truck carries a fixed number of drones to serve customers, and each drone can visit multiple customers per trip. However, the authors stipulate that only one drone can be launched or retrieved at the same node at any given time, even if a truck is armed with multiple drones. To solve the problem, the authors develop a drone truck route constructive heuristic and a large neighborhood search. 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 ALNS method and a two-stage matheuristic algorithm are developed in Chen et al. (2021a) and Chen et al. (2021b), respectively. Yu et al. (2020), Yu et al. (2022), and Bakach et al. (2021) also study the two-echelon delivery system with trucks and robots. Specifically, Yu et al. (2020) assume that trucks are only used to drop off and pick up robots at rendezvous nodes, and robots are responsible for serving customers. They first minimize the number of used trucks and then the total transportation costs of both types of vehicles. The authors develop a hybrid metaheuristic approach with backtracking to solve large-size instances. Yu et al. (2022) assume that both types of vehicles can serve customers directly. The authors state that specific parking nodes are available for trucks to rendezvous, replenish, and swap batteries for robots, i.e., these operations are not allowed to be conducted at customer nodes. They stipulate

that each truck can only and must carry one robot. They minimize the total travel cost and design an ALNS method. Bakach et al. (2021) consider a two-tier delivery system with trucks and local robot hubs. In their system, trucks travel at the first tier and drop off packages at second-tier robot hubs. A set of robots in each hub then deliver packages to final customers within their time windows. The robots can only carry one parcel per trip but can perform multiple trips. The authors minimize the number of hubs first and then the travel cost of robots. A commercial solver is used to solve models.

Table 1 A summary of exact solution methods for the truck-drone delivery system.

Authors	#T	#D	Flight range	Drone capacity	Drone allocation	Solution method	#O
Bouman et al. (2018)	1	1	Ignored	1		Dynamic programming	19
Poikonen et al. (2019)	1	1	Duration	1		Branch-and-bound	19
Vásquez et al. (2021)	1	1	Duration	1		Benders decomposition	19
El-Adle et al. (2021)	1	1	Duration	1		MILP and commercial solver	31
Roberti and Ruthmair (2021)	1	1	Energy related	1		Branch-and-price	39
Cavani et al. (2021)	1	m	Ignored	1		Branch-and-cut	24
Tamke and Buscher (2021)	m	m	Distance	1		Branch-and-cut	29
Wang and Sheu (2019)	m	m	Duration	m		Branch-and-price	13
Bakir and Tiniç (2020)	m	m	Duration	1		Dynamic discretization discovery	25
Zhen et al. (2022)	m	m	Duration	1		Branch-price-and-cut	14
Di Puglia Pugliese et al. (2021a)	m	m	Energy related	1		Benders decomposition	15
Salama and Srinivas (2020)	1	m	Distance	1		MILP and commercial solver	35
Kang and Lee (2021)	1	m	Energy related	1		Branch-and-cut	50
Dukkanci et al. (2021)	m	m	Energy related	1		SOCP and commercial solver	37
This paper	m	m	Energy related	1	✓	Branch-and-price	35

#T (#D): number of trucks (drones); m: multiple; #O: the maximal number of customers in an optimally solved instance.

From the papers reviewed above and Table 1, we find that almost all the studies considering multiple drones treat the number of drones in each truck as a given parameter. In fact, a delivery system often has a limited number of drones due to their high acquisition costs, suggesting that it is sometimes unpractical for each truck to arm multiple drones. Thus, this study complements current studies by considering the allocation decisions of drones to better plan delivery routes and serve customers.

2.2. Two-echelon Vehicle Routing Problem

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. Another significant difference is that the number of satellites in the 2E-VRP is often small. Most works using exact methods often set the number of satellites to about 5% of the number of customers. Thus, the first-echelon routes in the 2E-VRP can be enumerated or obtained by solving a TSP. The resolution process is then transformed into an enumeration procedure, a multi-depot VRP, and a combination procedure, significantly reducing the complexity of the problem (Santos et al. 2013, Dellaert et al. 2019, Santos et al. 2015, Marques et al. 2020). However, in the VRPD, each customer can be treated as a potential satellite or a second-echelon customer, making it impractical to enumerate all the routes for trucks as they may need to visit many customers.

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. An ALNS method is developed. For more details about the 2E-VRP, see the review paper by Li et al. (2021) and Sluijk et al. (2023).

3. Problem Definition and Formulation

This section defines the problem and constructs the mathematical models.

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 $Z = V \backslash \{0\}$ 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. We consider the deadline constraints because they are commonly imposed in some applications like the same-day delivery (Ulmer and Thomas 2018, Stroh et al. 2022). Moreover, Boysen et al. (2018) and Di Puglia Pugliese et al. (2021b) suggest that drones can deliver parcels without the presence of a customer, landing and releasing the package on a customer's balcony. Thus, time window restrictions for drone-served customers are not necessary. 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_i^d for each customer. $f_i^d = 1$ if customer i can be served by either a vehicle or a drone, and $f_i^d = 0$ if customer i can only be served by a vehicle. We denote $Z_d = \{i \mid f_i^d = 1, i \in Z\}$ as the set of customers that can be served by drones. A homogeneous fleet of vehicles $K_v = \{1, 2, ..., k_v\}$ and a homogeneous fleet of drones $K_d = \{1, 2, ..., k_d\}$ are ready at the depot to deliver parcels. The capacity of vehicles is Q, and each vehicle can carry at most Γ $(\Gamma \leq k_d)$ drones. Let q_1^d 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. For simplicity, we assume that vehicles and drones follow the same distance matrix. We note that the models and algorithms developed later in this study still apply when different distance matrices are defined. 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_i^v and s_i^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 $P(q) = (W + m + q)^{\frac{3}{2}} \sqrt{\frac{g^3}{2\rho\varsigma h}}$, 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), ρ is the fluid density of air (kg/m^3), and ς is the area of spinning blade disc (m^2). The unit of P is $Watt$. We set $q_2^d = W + m$, satisfying $q_2^d \leq q_1^d$, and let B_c 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 $P(q_j)t_{ij}^d + P(0)t_{ji}^d \leq B_c$ can be satisfied, $E_{ij} = 0$ otherwise. We note that with this pre-processing method, we can use any complex function to express drones' energy capacity constraints.

In the rest of this paper, we use the terms *vehicle node* and *drone node* to represent a customer 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 about the operations of trucks and drones: (i) Each drone can only be launched and retrieved by the same vehicle; (ii) 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; (iii) 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; (iv) Each drone trip consumes a

certain amount of energy. 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 Tiniç (2020), we assume the number of available batteries is sufficient.

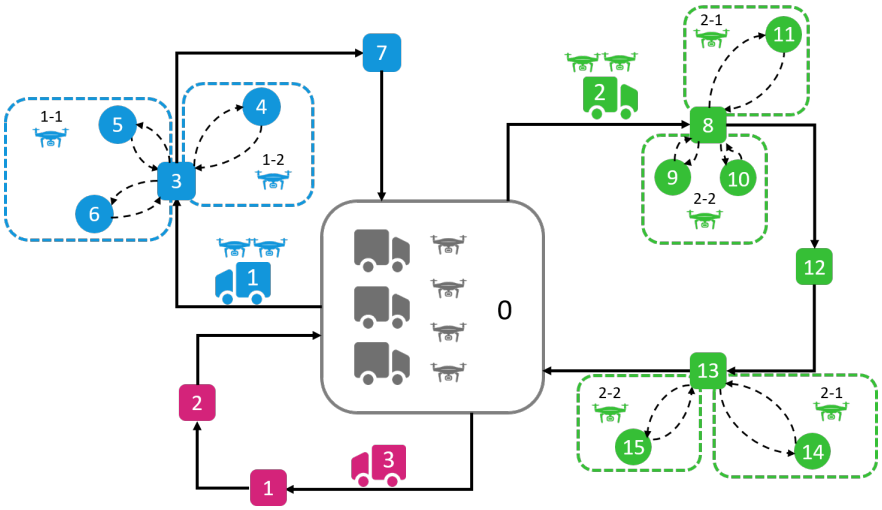


Figure 1 **An example of the 2E-VRP-D solution.**

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 to vehicle nodes. The waiting time of a vehicle is determined by its service time at a customer and the makespan of drone routes. Figure 1 shows a feasible solution to the 2E-VRP-D, where vehicle and drone nodes are denoted by squares and circles, respectively. There are three vehicles and four drones available at depot 0. Two drones are allocated to vehicles 1 and 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. An MILP Model

Based on the work of Chen et al. (2021a), we construct an MILP model for the 2E-VRP-D. We note that we have improved their formulation by reducing the index from 4 to 3, although our problem is more complex. 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;
- y_d^k: a binary variable that is 1 if vehicle $k \in K_v$ is equipped with drone $d \in K_d$, 0 otherwise;
- h_i^k: a binary variable that is 1 if vehicle $k \in K_v$ visiting customer $i \in Z$ is equipped with at least one drone, 0 otherwise;
- u_{ij}: a binary variable that is 1 if customer $j \in Z_d$ is served by a drone dispatched from 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 $i \in Z_d$ or customer $j \in Z_d$ is the first node of a drone route starting from node $i \in Z$, 0 otherwise;
- ω_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, ω_i^k would be 0;
- a_i: a continuous variable representing the arrival time of a vehicle or drone at node $i \in V$;
- φ_i: a continuous variable representing the waiting time of a vehicle at customer $i \in Z$.

The MILP model is constructed as follows:

$$\min \sum_{(i,j)\in A} \sum_{k\in K_v} t_{ij}^v x_{ij}^k + \sum_{i\in Z} \varphi_i \tag{1}$$

$$\text{s.t.} \sum_{i\in V} \sum_{k\in K_v} x_{ij}^k + \sum_{i\in Z} u_{ij} = 1 \qquad \forall j \in Z, \tag{2}$$

$$\sum_{j\in Z} x_{0j}^k \le 1 \qquad \forall k \in K_v, \tag{3}$$

$$\sum_{i\in V} x_{ij}^k = \sum_{i\in V} x_{ji}^k \qquad \forall j \in V, k \in K_v, \tag{4}$$

$$u_{ij} \le \sum_{k\in K_v} h_i^k \qquad \forall i \in Z, j \in Z_d, \tag{5}$$

$$h_i^k \le \sum_{d\in K_d} y_d^k \qquad \forall i \in Z, k \in K_v, \tag{6}$$

$$h_i^k \le \sum_{j\in V} x_{ji}^k \qquad \forall i \in Z, k \in K_v, \tag{7}$$

$$\sum_{d\in K_d} y_d^k \le \Gamma \qquad \forall k \in K_v, \tag{8}$$

$$u_{ij} \le E_{ij} \qquad \forall i \in Z, j \in Z_d, \tag{9}$$

$$\sum_{i\in Z} u_{ij} = \sum_{i\in Z} z_{ij} \qquad \forall j \in Z_d, \tag{10}$$

$$z_{ij} + \sum_{k\in K_v} h_i^k \le 1 + u_{ij} \qquad \forall i \in Z_d, j \in Z_d, \tag{11}$$

$$z_{ij} \le u_{ij} \qquad \forall i \in Z \setminus Z_d, j \in Z_d, \tag{12}$$

$$u_{li} - u_{lj} \le 1 + \sum_{k\in K_v} h_i^k - z_{ij} \qquad \forall i \in Z_d, j \in Z_d, l \in Z, \tag{13}$$

$$z_{ij} \le \sum_{l\in Z} u_{li} + \sum_{k\in K_v} h_i^k \qquad \forall i \in Z_d, j \in Z_d, \tag{14}$$

$$\sum_{j\in Z_d} z_{ij} \le \sum_{d\in K_d} y_d^k + M(1 - h_i^k) \qquad \forall i \in Z, k \in K_v, \tag{15}$$

$$\sum_{j\in Z_d} z_{ij} \le 1 + M' \sum_{k\in K_v} h_i^k \qquad \forall i \in Z, \tag{16}$$

$$q_j + \sum_{i\in Z_d} q_i u_{ji} - \omega_j^k \le M_j(1 - \sum_{i\in V} x_{ij}^k) \qquad \forall j \in Z, k \in K_v, \tag{17}$$

$$\sum_{j\in Z} \omega_j^k + q_1^d \sum_{d\in K_d} y_d^k \le Q \qquad \forall i \in Z, k \in K_v, \tag{18}$$

$$t_{0j}^v - a_j \le M_j'(1 - \sum_{k\in K_v} x_{0j}^k) \qquad \forall j \in Z, \tag{19}$$

$$a_i + \varphi_i + t_{ij}^v - a_j \leq M_{ij}(1 - \sum_{k \in K_v} x_{ij}^k) \qquad \forall i \in Z, j \in V, \tag{20}$$

$$a_i + t_0 + t_{ij}^d - a_j \leq M_{ij}^{'}(2 - z_{ij} - u_{ij}) \qquad \forall i \in Z, j \in Z_d, \tag{21}$$

$$a_i + s_i^d + t_{il}^d + t_0 + t_{lj}^d - a_j \leq M_{ijl}(2 - z_{ij} - u_{li}) \qquad \forall l \in Z, i \in Z_d, j \in Z_d, \tag{22}$$

$$\varphi_i \geq s_i^v \sum_{j \in V} \sum_{k \in K_v} x_{ji}^k \qquad \forall i \in Z, \tag{23}$$

$$a_j + s_j^d + t_{ji}^d - a_i - \varphi_i \leq M_{ij}^{''}(1 - u_{ij}) \qquad \forall i \in Z, j \in Z_d, \tag{24}$$

$$a_i \leq l_i \qquad \forall i \in V, \tag{25}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, k \in K_v, \tag{26}$$

$$y_d^k \in \{0,1\} \qquad \forall k \in K_v, d \in K_d, \tag{27}$$

$$h_i^k,\ u_{ij},\ z_{ij} \in \{0,1\} \qquad \forall i \in Z, j \in Z_d, k \in K_v, \tag{28}$$

$$\omega_i^k \geq 0 \qquad \forall i \in Z, k \in K_v, \tag{29}$$

$$a_i \geq 0 \qquad \forall i \in V, \tag{30}$$

$$\varphi_i \geq 0 \qquad \forall i \in Z. \tag{31}$$

The objective function (1) minimizes the total duration of all routes. Constraints (2) ensure that each customer is visited by a vehicle or a drone exactly once. Constraints (3) mean that each vehicle can leave the depot at most once. Constraints (4) ensure the flow conservation at each node. Constraints (5)–(7) impose that drones can only be launched from customers where a vehicle stops. Constraints (8) restrict that a vehicle can carry at most Γ drones. Constraints (9) mean that we can use a drone to serve a customer only when it is energy feasible. Constraints (10) suggest that a drone will serve node j or not depending on whether j belongs to a drone route. Constraints (11) and (12) characterize the case where j is the first drone node in a drone route. Specifically, constraints (11) mean that if $z_{ij} = 1$ and $\sum_{k \in K_v} h_i^k = 1$, indicating that node i is a vehicle node and is followed by a drone node j, then u_{ij} must be 1. Constraints (12) mean that if node j is visited by a drone dispatched from a vehicle node i, then u_{ij} must be 1. Constraints (13) denote the case where node j is visited by a drone dispatched from a vehicle node l but not the first drone node of a drone route. Let i be a drone node visited before j, then we have $u_{li} = 1$, $\sum_{k \in K_v} h_i^k = 0$, and $z_{ij} = 1$, imposing that u_{lj} must be 1. Constraints (14) define variable z_{ij}, i.e., it can be 1 only if $\sum_{l \in Z} u_{li} = 1$ or $\sum_{k \in K_v} h_i^k = 1$. Constraints (15) impose that the number of drones dispatched from a customer location cannot exceed the number of drones carried by the associated vehicle. We set the large constant $M = |Z_d|$. Constraints (16) denote that if i is not a vehicle node, the number of arcs leaving i is at most 1. We set the large constant $M' = |Z_d| - 1$. Constraints (17) calculate the value of ω_j^k, and we set the large constant $M_j = q_j + \sum_{i \in Z_d} q_i$. Inequalities (18) ensure that vehicle

capacity constraints are respected. The total payload of a vehicle includes the demands of customers and the weights of drones and their support equipment. Constraints (19)–(22) calculate the arrival times at customers. We set the large constants $M'_j = t^v_{0j}$, $M_{ij} = l_0 + t^v_{ij}$, $M'_{ij} = l_i + t_0 + t^d_{ij}$, and $M_{ijl} = l_0 + s^d_i + t^d_{il} + t_0 + t^d_{lj}$. Constraints (23) and (24) 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 constant $M''_{ij} = l_0 + s^d_j + t^d_{ji}$. Inequalities (25) impose the deadline constraints. Constraints (26)–(31) define the domains of decision variables.

3.3. A Set Partitioning Formulation

Although the MILP model (1)–(31) can be directly tackled by off-the-shelf solvers like CPLEX and Gurobi, solvers can usually not provide good solutions for medium- or large-size instances in a reasonable computing time because the 2E-VRP-D is NP-hard—it reduces to the VRP when $K_d = \emptyset$, which is a well-known NP-hard problem. To solve instances more efficiently, this section introduces a set partitioning model, based on which a B&P algorithm is developed in the next section.

Let $\mathcal{R}$ be the set of all feasible routes and c_r be the shortest duration of route $r \in \mathcal{R}$. Parameter α_{ir} is the number of times customer $i \in Z$ is visited by route $r \in \mathcal{R}$. Parameter d_r is the number of drones allocated to route $r \in \mathcal{R}$. We define μ_r as a binary variable equalling to 1 if route $r \in \mathcal{R}$ is selected, and 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 Γ; (iv) The vehicle capacity constraint, drone energy capacity constraint, and customer deadline constraints are respected.

The set partitioning formulation for the 2E-VRP-D is constructed as follows:

$$\min \sum_{r\in\mathcal{R}} c_r \mu_r \tag{32}$$

$$\text{s.t.} \sum_{r\in\mathcal{R}} \alpha_{ir} \mu_r = 1 \qquad \forall i \in Z, \tag{33}$$

$$\sum_{r\in\mathcal{R}} \mu_r \le k_v, \tag{34}$$

$$\sum_{r\in\mathcal{R}} d_r \mu_r \le k_d, \tag{35}$$

$$\mu_r \in \{0,1\} \qquad \forall r \in \mathcal{R}. \tag{36}$$

The objective function (32) minimizes the total duration of selected routes. Constraints (33) force each customer to be visited exactly once. Constraints (34)–(35) limit the number of available vehicles and drones, respectively. Constraints (36) define the domains of variables.

4. Branch-and-Price Algorithm

Due to the large size of set $\mathcal{R}$, it is impractical to enumerate all the feasible routes. Thus, we use a CG algorithm to solve the linear relaxation problem (LRP) at each node of the B&B tree, leading to

a B&P algorithm. Figure 2 shows the algorithm framework. The B&P starts with a subset $\mathcal{R}' \subseteq \mathcal{R}$ of initial columns created by the cheapest insertion heuristic (see Section 4.2). Then, each node in the B&B tree is explored. For a selected node, the CG algorithm is used. At each iteration, the CG solves a restricted master problem (RMP) by considering the LRP with $\mathcal{R}'$ and produces the dual solution, passed to the pricing problem. Subsequently, a tabu search heuristic (see Section 4.1.4) is applied to solve the pricing problem to generate columns with negative reduced costs. If the number of generated columns does not reach the limit col_{max}, the exact labeling algorithm (see Sections 4.1.1–4.1.3) will be applied by respectively assuming that $\bar{k}_d = 0, \ldots, \Gamma$ drones are carried by a vehicle until col_{max} columns are found. All new columns are added to set $\mathcal{R}'$, and the RMP is solved again. The CG procedure terminates when no columns with negative reduced costs can be found. If the optimal solution (extended from the optimal solution of the current RMP) of the LRP is integral, we update the upper bound and check the algorithm termination condition; otherwise, we branch on the fractional solution, update the lower bound, and choose a new tree node to explore.

4.1. The Pricing Problem

Let $\lambda_i, i \in Z$ be the dual variables associated with constraints (33). Let λ_0^v ($\lambda_0^v \leq 0$) and λ_0^d ($\lambda_0^d \leq 0$) be the dual variables associated with constraints (34) and (35), respectively. The reduced cost of route $r \in \mathcal{R}$ is

$$\bar{c}_r = c_r - \sum_{i \in Z} \alpha_{ir} \lambda_i - \lambda_0^v - d_r \lambda_0^d. \tag{37}$$

The purpose of the pricing problem is to find routes with negative reduced costs. For the 2E-VRP-D, 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 $\Gamma + 1$ times, each time with a different number of associated drones. For example, if each vehicle can carry at most $\Gamma = 3$ drones, we then run the labeling algorithm four times, assuming that $0, 1, 2$, and 3 drones are carried, respectively. Note that we use $\bar{k}_d$ to represent the number of drones carried by a vehicle in the current labeling procedure. The labeling algorithm stops when col_{max} columns with negative reduced costs are found. Our preliminary tests show that $col_{max} = 10$ is sufficient to provide satisfactory results.

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

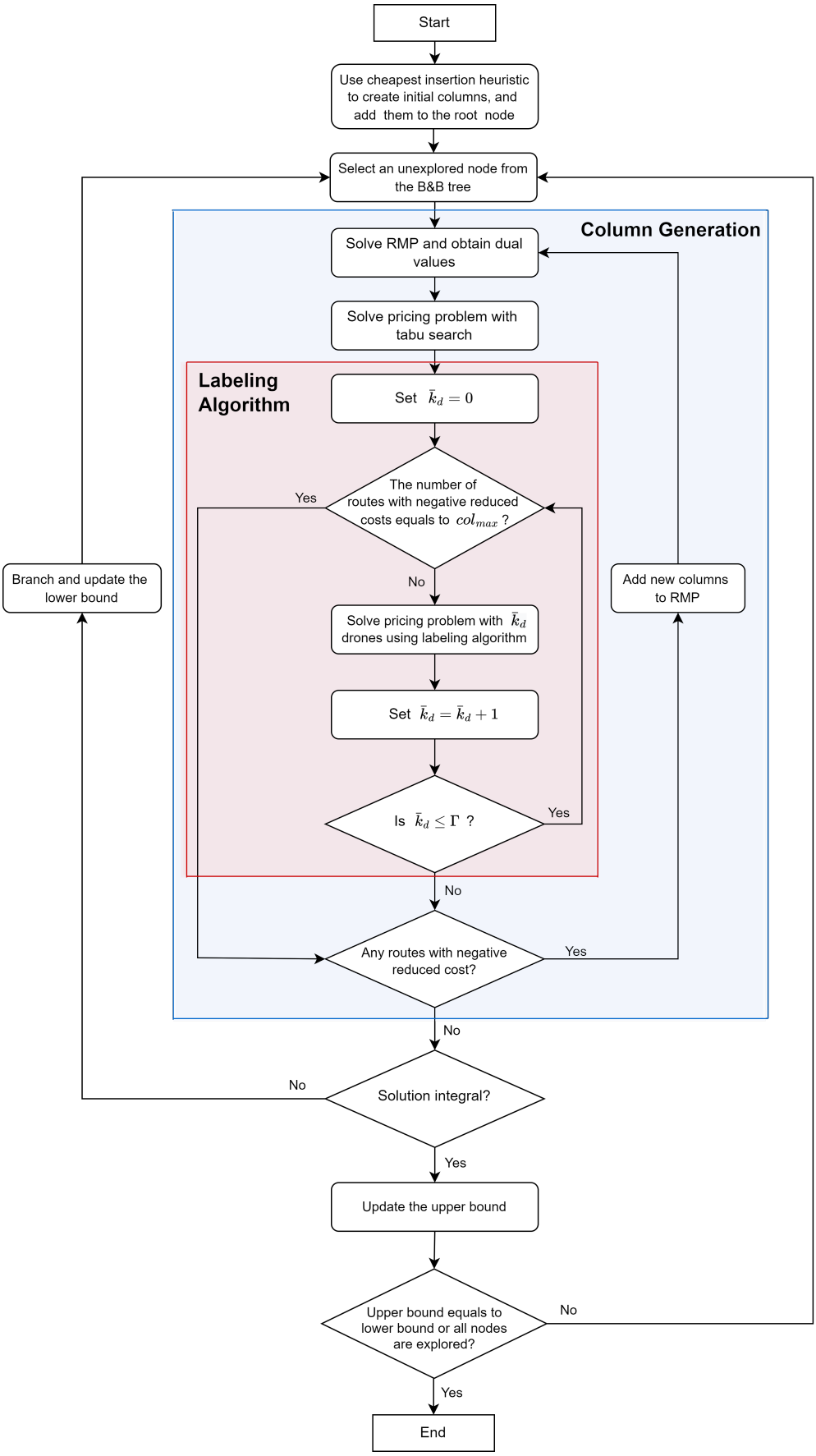


Figure 2 **The flowchart of the B&P algorithm.**

- $v(L_f)$: 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 $p(L_f)$.
- $\pi^i(L_f)$: the time of drone i $(i = 1, \ldots, \bar{k}_d)$ returning to $\sigma(L_f)$ after serving all assigned customers. If $i = 0$, $\pi^i(L_f)$ 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 λ_0^v and λ_0^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 $L_f = (0, 0, \mathbf{0}, Q - q_1^d \bar{k}_d, 0, V, Z_d)$. 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 $\max_{j=0,\ldots,\bar{k}_d}\{\pi^j(L'_f)\} + t^v_{\sigma(L'_f)w} \leq l_w$. The following relations are applied to obtain the attributes of label L_f.

- $v(L_f) = w$.
- $\sigma(L_f) = w$.
- $\pi^i(L_f) = \begin{cases} \max\limits_{j=0,\ldots,\bar{k}_d} \{\pi^j(L'_f)\} + t^v_{\sigma(L'_f)w} & \text{if } i = 1, \ldots, \bar{k}_d, \\ \max\limits_{j=0,\ldots,\bar{k}_d} \{\pi^j(L'_f)\} + t^v_{\sigma(L'_f)w} + s^v_w & \text{if } i = 0. \end{cases}$
- $\kappa(L_f) = \kappa(L'_f) - q_w$.
- $C(L_f) = C(L'_f) + \lambda_w$.
- $\Omega_1(L_f) = \Omega_1(L'_f) \setminus \left\{w' : (w, w') \in A, \pi^0(L_f) + t^v_{ww'} > l_{w'} \text{ or } \kappa(L_f) - q_{w'} < 0\right\} \setminus \{w\}$.
- $\Omega_2(L_f) = \Omega_2(L'_f) \setminus \left\{w' : w' \in Z_d, \pi^i(L_f) + t_0 + t^d_{ww'} > l_{w'}, i \neq 0 \text{ or } \kappa(L_f) - q_{w'} < 0\right\} \setminus \{w\}$.

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 $(i = 1, \ldots, \bar{k}_d)$ such that $\pi^i(L'_f) + t_0 + t^d_{\sigma(L'_f)w} \leq l_w$. The following relations are applied to generate label L_f.

- $v(L_f) = w$.
- $\sigma(L_f) = \sigma(L'_f)$.
- $\pi^i(L_f) = \begin{cases} \pi^i(L'_f) & \text{if } i = 0, \\ \pi^i(L'_f) & \text{if drone } i \ (i = 1, \ldots, \bar{k}_d) \text{ does not visit } w, \\ \pi^i(L'_f) + t_0 + t^d_{\sigma(L'_f)w} + s^d_w + t^d_{w\sigma(L'_f)} & \text{if drone } i \ (i = 1, \ldots, \bar{k}_d) \text{ visits } w. \end{cases}$
- $\kappa(L_f) = \kappa(L'_f) - q_w$.

- $C(L_f) = C(L'_f) + \lambda_w$.
- $\Omega_1(L_f) = \Omega_1(L'_f) \setminus \{w' : (w, w') \in A, \max\limits_{i=0,\dots,\bar{k}_d} \{\pi^i(L_f)\} + t^v_{\sigma(L_f)w'} > l_{w'} \text{ or } \kappa(L_f) - q_{w'} < 0\} \setminus \{w\}$.
- $\Omega_2(L_f) = \Omega_2(L'_f) \setminus \{w' : w' \in Z_d, \min\limits_{i=1,\dots,\bar{k}_d} \{\pi^i(L_f)\} + t_0 + t^d_{\sigma(L_f)w'} > l_{w'} \text{ or } \kappa(L_f) - q_{w'} < 0\} \setminus \{w\}$.

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 Case 1 only applies when $vel^d \geq vel^v$, otherwise the triangle inequality of travel time $d_{ij}/vel^d \leq d_{ik}/vel^v + \min\{s^v_k + d_{kj}/vel^v, d_{kj}/vel^d\}$ may not hold. Fortunately, the relation $vel^d \geq vel^v$ 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 $\Xi(L_f)$ 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 $p^b \in \Xi(L_f)$ be one of the partial paths and $\mathcal{R}(p(L_f) \oplus p^b)$ 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) $\Xi(L^2_f) \subseteq \Xi(L^1_f)$,

(2) $\min\limits_{p_1 \in \mathcal{R}(p(L^1_f) \oplus p^b)} \{\bar{C}(p_1)\} \leq \min\limits_{p_2 \in \mathcal{R}(p(L^2_f) \oplus p^b)} \{\bar{C}(p_2)\} \quad \forall p^b \in \Xi(L^2_f)$.

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 *$L_f^1 \prec L_f^2$ if the following conditions are satisfied*

$$\sigma(L_f^1) = \sigma(L_f^2), \tag{38}$$

$$\pi^i(L_f^1) \leq \pi^i(L_f^2) \qquad i = 0, \ldots, \bar{k}_d, \tag{39}$$

$$\kappa(L_f^1) \geq \kappa(L_f^2), \tag{40}$$

$$\Omega_1(L_f^2) \subseteq \Omega_1(L_f^1), \tag{41}$$

$$\Omega_2(L_f^2) \subseteq \Omega_2(L_f^1), \tag{42}$$

$$\pi^i(L_f^1) - C(L_f^1) - \pi^i(L_f^2) + C(L_f^2) \leq 0 \quad i = 0, \ldots, \bar{k}_d. \tag{43}$$

This dominance rule does not require that $v(L_f^1) = v(L_f^2)$, which extends the space for performing dominance tests. Conditions (38)–(42) ensure that $\Xi(L_f^2) \subseteq \Xi(L_f^1)$ and conditions (43) 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 3 shows an example, where vehicle 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 (39) and (43).

Proof of Proposition 1: See Appendix A.1. □

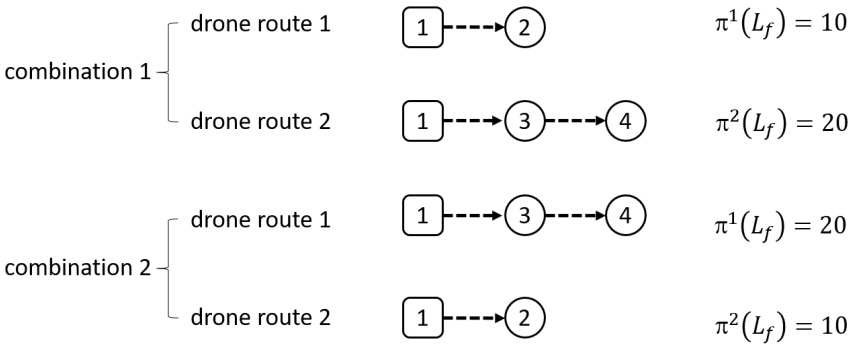


Figure 3 An example of drone route exchange.

To check if relations (39) and (43) are satisfied, we may need to compare (at most) $\bar{k}_d!$ times for all combinations of $\pi^i(L_f^1)$ and $\pi^i(L_f^2)$. As both relations involve the calculation of $\pi^i(L_f^1) - \pi^i(L_f^2)$ for all $i = 0, \ldots, \bar{k}_d$, under given combinations we only need to find the value of $\max_{i=0,\ldots,\bar{k}_d}\{\pi^i(L_f^1) -$

$\pi^i(L_f^2)\}$ and check if it satisfies the relations. If the minimum of $\max_{i=0,\dots,\bar{k}_d}\{\pi^i(L_f^1) - \pi^i(L_f^2)\}$ satisfy conditions (39) and (43) and all other conditions are also met, we can say $L_f^1 \prec L_f^2$. To help understand this statement, an example is given as follows. We assume that two labels L_f^1 and L_f^2 satisfy constraints (38), (40)–(42) and $\bar{k}_d = 3$ drones are used. Meanwhile, $\pi^1(L_f^1) = 15, \pi^2(L_f^1) = 10, \pi^3(L_f^1) = 7$ and $\pi^1(L_f^2) = 9, \pi^2(L_f^2) = 13, \pi^3(L_f^2) = 17$. Note that this is only one combination of $\pi^i(L_f^1)$ and $\pi^i(L_f^2)$, and each of them has a total of $\bar{k}_d! = 3! = 6$ combinations. Constraints (39) and (43) can be converted to $\max_{i=0,\dots,\bar{k}_d}\{\pi^i(L_f^1) - \pi^i(L_f^2)\} \leq \max\{0, C(L_f^1) - C(L_f^2)\}$. For the given combinations, $\max_{i=0,\dots,\bar{k}_d}\{\pi^i(L_f^1) - \pi^i(L_f^2)\} = 6 > 0$, thus they do not meet the dominance rule. However, if we exchange $\pi^1(L_f^1)$ with $\pi^3(L_f^1)$, i.e., let $\pi^1(L_f^1) = 7$ and $\pi^3(L_f^1) = 15$, we will have a smaller value of $\max_{i=0,\dots,\bar{k}_d}\{\pi^i(L_f^1) - \pi^i(L_f^2)\}$, which equals $\pi^2(L_f^1) - \pi^2(L_f^2) = -3 < 0$, satisfying the constraints. The problem is then transformed to find one pair of combinations of $\pi^i(L_f^1)$ and $\pi^i(L_f^2)$ minimizing $\max_{i=0,\dots,\bar{k}_d}\{\pi^i(L_f^1) - \pi^i(L_f^2)\}$.

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

Proof of Corollary 1: See Appendix A.2. □

Using Corollary 1, we can sort $\pi^i(L_f)$ and $\pi^j(L_f)$ before applying dominance rules. 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* $i' = \text{argmin}_{i=1,\dots,\bar{k}_d}\{\pi^i(L_f)\}$ *visits this new node, which will not affect the optimality of the partial route.*

Proof of Proposition 2: See Appendix A.3. □

4.1.2. Backward Labeling. Let $p(L_b)$ be a feasible backward partial path, which is denoted by a label $L_b = (v(L_b), \sigma(L_b), \pi^i(L_b), \rho^i(L_b), \kappa(L_b), C_1(L_b), C_2(L_b), \Omega_1(L_b), \Omega_2(L_b))$, where attributes $v(L_b)$, $\sigma(L_b)$, $\kappa(L_b)$, $\Omega_1(L_b)$, and $\Omega_2(L_b)$ 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 ($i = 1, \dots, \bar{k}_d$) 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 ($i = 1, \dots, \bar{k}_d$); 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 $L_b = (0, 0, l_0, \mathbf{0}, Q - q_1^d \bar{k}_d, 0, 0, V, Z_d)$. Given a label L'_b, 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 $\min_{i=0,\ldots,\bar{k}_d}\{\pi^i(L'_b)\} - t^v_{w\sigma(L'_b)} - s^v_w \geq 0$. The following relations are used to obtain the attributes of label L_b.

- $v(L_b) = w$.
- $\sigma(L_b) = w$.
- $\pi^i(L_b) = \begin{cases} \min\limits_{j=0,\ldots,\bar{k}_d}\{\pi^j(L'_b)\} - t^v_{w\sigma(L'_b)} & \text{if } i = 1,\ldots,\bar{k}_d, \\ \min\left\{\min\limits_{j=0,\ldots,\bar{k}_d}\{\pi^j(L'_b)\} - t^v_{w\sigma(L'_b)} - s^v_w, l_w\right\} & \text{if } i = 0. \end{cases}$
- $\rho^i(L_b) = \begin{cases} 0 & \text{if } i = 1,\ldots,\bar{k}_d, \\ s^v_w & \text{if } i = 0. \end{cases}$
- $\kappa(L_b) = \kappa(L'_b) - q_w$.
- $C_1(L_b) = C_1(L'_b) + \lambda_w$.
- $C_2(L_b) = C_2(L'_b) + \max\limits_{i=0,\ldots,\bar{k}_d}\{\rho^i(L'_b)\} + t^v_{w\sigma(L'_b)}$.
- $\Omega_1(L_b) = \Omega_1(L'_b) \setminus \{w' : (w, w') \in A \text{ and } \kappa(L_b) - q_{w'} < 0\} \setminus \{w\}$.
- $\Omega_2(L_b) = \Omega_2(L'_b) \setminus \{w' : w' \in Z_d \text{ and } \kappa(L_b) - q_{w'} < 0\} \setminus \{w\}$.

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 $\min\{\pi^i(L'_b) - t^d_{w\sigma(L'_b)} - s^d_w, l_w\} - t^d_{\sigma(L'_b)w} - t_0 \geq 0$. The following relations are used to obtain the attributes of label L_b.

- $v(L_b) = w$.
- $\sigma(L_b) = \sigma(L'_b)$.
- $\pi^i(L_b) = \begin{cases} \pi^i(L'_b) & \text{if } i = 0, \\ \pi^i(L'_b) & \text{if drone } i \ (i = 1,\ldots,\bar{k}_d) \text{ does not visit } w, \\ \min\{\pi^i(L'_b) - t^d_{w\sigma(L'_b)} - s^d_w, l_w\} - t^d_{\sigma(L'_b)w} - t_0 & \text{if drone } i \ (i = 1,\ldots,\bar{k}_d) \text{ visits } w. \end{cases}$
- $\rho^i(L_b) = \begin{cases} \rho^i(L'_b) & \text{if } i = 0, \\ \rho^i(L'_b) & \text{if drone } i \ (i = 1,\ldots,\bar{k}_d) \text{ does not visit } w, \\ \rho^i(L'_b) + t^d_{\sigma(L'_b)w} + s^d_w + t^d_{w\sigma(L'_b)} + t_0 & \text{if drone } i \ (i = 1,\ldots,\bar{k}_d) \text{ visits } w. \end{cases}$
- $\kappa(L_b) = \kappa(L'_b) - q_w$.
- $C_1(L_b) = C_1(L'_b) + \lambda_w$.
- $C_2(L_b) = C_2(L'_b)$.
- $\Omega_1(L_b) = \Omega_1(L'_f) \setminus \{w' : (w, w') \in A \text{ and } \kappa(L_b) - q_{w'} < 0\} \setminus \{w\}$.
- $\Omega_2(L_b) = \Omega_2(L'_f) \setminus \{w' : w' \in Z_d \text{ and } \kappa(L_b) - q_{w'} < 0\} \setminus \{w\}$.

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 *$L_b^1 \prec L_b^2$ if the following conditions are satisfied*

$$\begin{aligned}
&\sigma(L_b^1) = \sigma(L_b^2),\\
&\pi^i(L_b^1) \geq \pi^i(L_b^2) \quad i = 0,\ldots,\bar{k}_d,\\
&\kappa(L_b^1) \geq \kappa(L_b^2),\\
&\Omega_1(L_b^2) \subseteq \Omega_1(L_b^1),\\
&\Omega_2(L_b^2) \subseteq \Omega_2(L_b^1),\\
&\rho^i(L_b^1) + C_2(L_b^1) - C_1(L_b^1) - \rho^i(L_b^2) - C_2(L_b^2) + C_1(L_b^2) \leq 0 \quad i = 0,\ldots,\bar{k}_d.
\end{aligned}$$

Proof of Proposition 3: See Appendix A.4. □

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 $\pi^i(L_b)$ and $\rho^i(L_b)$, 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.

4.1.3. Label Joining. When $\bar{k}_d = 0$, a forward label and a backward label can be extended only when $\pi^0(L_f) - s^v_{\sigma(L_f)} < l_0/2$ and $\pi^0(L_b) > l_0/2$, 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:

$$\sigma(L_f) = \sigma(L_b), \tag{44}$$

$$\pi^i(L_f) \leq \pi^i(L_b) \quad i = 0,\ldots,\bar{k}_d, \tag{45}$$

$$\kappa(L_f) + \kappa(L_b) + q_{\sigma(L_f)} + q_1^d \bar{k}_d \geq Q, \tag{46}$$

$$S(L_f) \cap S(L_b) = \emptyset, \tag{47}$$

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 (44) ensures that the two partial paths end at the same vehicle node. Conditions (45)–(47) 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

$$\bar{c}_r = c_r - C(L_f) - C_1(L_b) - \lambda_0^v - \bar{k}_d \lambda_0^d.$$

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 $\pi^i(L_f) \leq \pi^j(L_b)$ 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 $\rho^j(L_b)$. 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 1.

Algorithm 1 The pseudo code of the pairing algorithm.

Input: forward label L_f and backward label L_b.
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.
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 $\rho^i(L_b)$.
for all route $f \in \mathcal{F}$ **do**
  for all route $b \in \mathcal{B}$ **do**
    if $\pi(f) \leq \pi(b)$ **then**
      $P \leftarrow P \cup \{(f, b)\}$, $\mathcal{B} \leftarrow \mathcal{B} \setminus \{b\}$;
      break;
Output: set P.

Corollary 2 *Algorithm 1 can always obtain the best pairs of partial drone routes.*

Proof of Corollary 2: See Appendix A.5. □

4.1.4. 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: *insertion*, *removal*, and *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

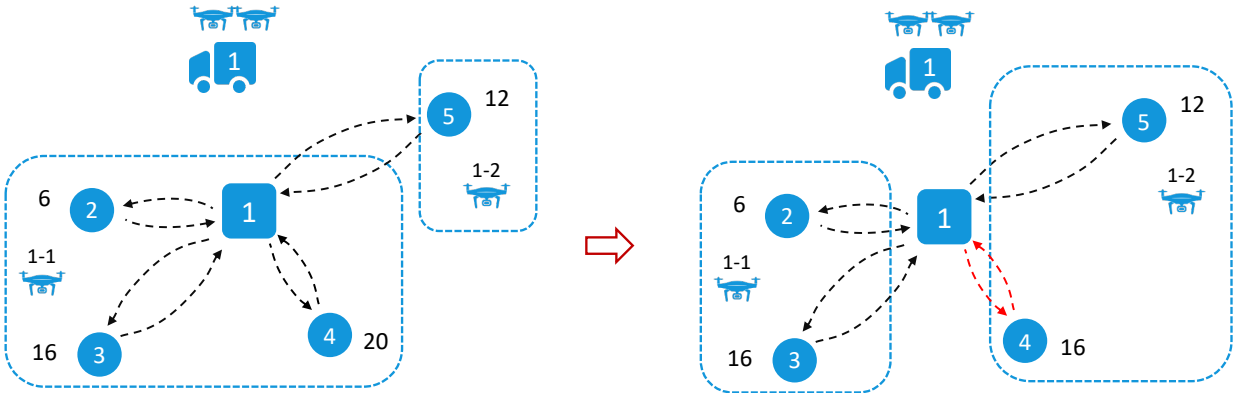


Figure 4 **An example of the shift operator used in the TS algorithm.**

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 4 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). Based on some preliminary tests, we set the maximum iteration of TS to 100.

4.2. Cheapest Insertion Heuristic for Initial Columns

We use the cheapest insertion heuristic to generate initial columns. The pseudo-code is provided in Algorithm 2. 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 Φ 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 Φ in non-ascending order concerning the value of $|\phi(i)|$. Next, we treat the first node i' in Φ 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

Algorithm 2 The pseudo code of the cheapest insertion heuristic.

Input: customer set Z, vehicle number k_v, and drone number k_d.
Define Φ: the set of unserved customers; $\phi(i)$: the set of unserved customers reachable by a drone from $i \in \Phi$.
Initialize $\Phi \leftarrow Z$; $\mathcal{R}' \leftarrow k_v$ empty routes, each with $\lceil k_d/k_v \rceil$ or $\lfloor k_d/k_v \rfloor$ drones.
while $\Phi \neq \emptyset$ **do**
  for all $i \in \Phi$ **do**
    $\phi(i) \leftarrow \{j : E_{ij} = 1, j \in \Phi\}$.
  Sort Φ in non-ascending order concerning the value of $|\phi(i)|$.
  $i' \leftarrow$ the first node in Φ; $\delta_{min} \leftarrow +\infty$.
  for all route $r \in \mathcal{R}'$ **do**
    for all insertion position pos in route r **do**
      if position pos is feasible for inserting i' **then**
        Calculate the incremental time δ after inserting i' into pos of route r.
        if $\delta < \delta_{min}$ **then**
          $\delta_{min} = \delta, r^* = r, pos^* = pos$.
  Insert customer i' into position pos^* of route r^*; $\Phi \leftarrow \Phi \setminus \{i'\}$.
  Sort $\phi(i')$ in non-descending order concerning customers' service deadlines.
  for all $j \in \phi(i')$ **do**
    Select the drone route dr with the minimum completion time.
    if the end of dr is feasible for inserting j **then**
      Insert j into the end of route dr; $\Phi \leftarrow \Phi \setminus \{j\}$.
Output: route set $\mathcal{R}'$.

node in $\phi(i')$ to the drone route with the minimum completion time. If the insertion operation is feasible, we remove this node from Φ. The procedures described above are repeated until $\Phi = \emptyset$.

Due to the limited number of vehicles and the deadline constraints, sometimes the heuristic may not provide a feasible solution. When a customer i cannot be inserted into any existing route, we add a dummy variable (say χ_i) associated with this customer into constraints (33) and a term $M\chi_i$ into the objective function. The dummy variables will guarantee the feasibility of an initial solution, and the large constant M will make these variables zero in the optimal solution. For the existing routes generated by the heuristic, we add them into $\mathcal{R}'$ as initial columns.

4.3. 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 $\widetilde{H}_v = \sum_{r\in\mathcal{R}} \widetilde{\mu}_r$ represent the number of vehicles used in the optimal solution of the LRP. If $\widetilde{H}_v$ is fractional, we create two child nodes with constraints $\sum_{r\in\mathcal{R}} \mu_r \leqslant \lfloor \widetilde{H}_v \rfloor$ and $\sum_{r\in\mathcal{R}} \mu_r \geqslant \lceil \widetilde{H}_v \rceil$, respectively.

(2) *The number of used drones*. Let $\widetilde{H}_d = \sum_{r\in\mathcal{R}} d_r \widetilde{\mu}_r$ represent the number of drones used in the optimal solution of the LRP. If $\widetilde{H}_d$ is fractional, we create two child nodes with constraints $\sum_{r\in\mathcal{R}} d_r \mu_r \leqslant \lfloor \widetilde{H}_d \rfloor$ and $\sum_{r\in\mathcal{R}} d_r \mu_r \geqslant \lceil \widetilde{H}_d \rceil$, respectively.

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

(4) *The flow between a vehicle node and a drone node*. Let $\widetilde{u}_{ij} = \sum_{r\in\mathcal{R}} \xi_{ijr} \widetilde{\mu}_r$ be the total flow between vehicle node i and drone node j, where parameter ξ_{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 $\widetilde{u}_{i^*j^*}$ is the closest to 0.5. Two child nodes are created by adding constraints $\sum_{r\in\mathcal{R}} \xi_{i^*j^*r} \mu_r \leqslant \lfloor \widetilde{u}_{i^*j^*} \rfloor$ and $\sum_{r\in\mathcal{R}} \xi_{i^*j^*r} \mu_r \geqslant \lceil \widetilde{u}_{i^*j^*} \rceil$.

5. Computational Experiments

This section presents the instance sets and analyzes the performance of the B&P. 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 2.5GHz Intel(R) Xeon(R) Platinum 8163HB CPU and 16G of memory under the Windows 10 operating system. The time limit on each run of the B&P 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 use their instances with 15 and 25 customers and further generate instances with 10 customers by removing the last five customers of the 15-customer instances. We name instances in this set *Cardiff_n_o*, where n denotes the number of customers and o is the order of the instance. The second instance set is based on the well-known Solomon (1987) instances. We generate instances by adopting the data of the first 35 customers of their type C instances and name them *c10X_n*, where *c*10*X* is the original instance name in Solomon (1987) and

n is the number of customers. Considering the capacity of our drones, we set $f_i^d = 0$ if $q_i > 10\ kg$ for both data set, i.e., if the demand of a customer is larger than 10 kg, it cannot be visited by a drone. As the number of customers satisfying $q_i > 10\ kg$ is quite small in the first data set, we further randomly select $\lfloor 0.2n \rfloor$ customers from the rest customers and set them as drone-forbidden customers, to simulate other cases where drone delivery is infeasible. We define 10- and 15-customer instances as small-size instances and 25- and 35-customer instances as medium-size instances. For 10- and 15-customer instances, we set $Q = 170\ kg$ and $\Gamma = 2$. For 25- and 35-customer instances, we respectively set $Q = 200\ kg$ and 250 kg, and let $\Gamma = 3$. As in Chen et al. (2021a), we set $s_i^d = \lfloor 0.5 s_i^v \rfloor, i \in Z$ for all the instances.

We let $g = 9.81\ N/kg$ and $\rho = 1.204\ kg/m^3$ 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 (SF Technology n.d.) to perform short- to mid-range deliveries. Specifically, according to the website, we set $q_d^2 = 37\ kg$, $vel^d = 15\ m/s$, and $h = 8$. The values of parameters q_d^1, ς, and B_c are not given on the website. Thus, we estimate $q_d^1 = 50\ kg$ based on the assumption that $q_d^1 \geq q_d^2$ and $\varsigma = 0.025\ m^2$ based on the picture of the drone as shown in Figure 5. Since we cannot find any information on the battery used by the Ark Octocopter Drone from the website, we set the battery capacity according to Stolaroff et al. (2018), who use LiPo batteries in their experiments. The authors state that the energy density of the LiPo batteries is 540 kJ/kg (i.e., 0.15 kWh/kg) and the battery weight of their octocopter is 4.2 kg. Thus we set $B_c = 0.15\ kWh/kg \times 4.2\ kg = 0.63 kWh$ for the first instance set. When using this battery capacity for the second instance set, we find that our algorithm cannot solve any instance to optimality within the time limit. This is because the travel times among customers in the second instance set are much shorter than those in the first one. Specifically, the average travel time in the first instance set is 162.73 seconds, whereas it is only 17.62 seconds in the second one. The shorter travel times deactivate drones' energy constraints, i.e., all the customers are eligible to be served by a drone sent out from any location. To better test the effectiveness of our algorithm, we consider another type of battery with weight 0.26 kg suggested in Stolaroff et al. (2018) and set $B_c = 0.15\ kWh/kg \times 0.26\ kg = 0.039\ kWh$ for the second instance set. We set vehicle speed $vel^v = 12\ m/s$ according to Moshref-Javadi et al. (2020).

5.2. Algorithm Performance

In this section, we first use small-size instances to evaluate the performance of the B&P by comparing it with the MILP model solved by CPLEX (Section 5.2.1). We then explore the performance of the B&P for medium-size instances in Section 5.2.2. Finally, we compare the TS algorithm for heuristic column generation and the exact labeling algorithm in Section 5.2.3. Note that both the TS and the exact labeling algorithm are used to solve the pricing problems for the experiments in

Figure 5 **The Ark Octocopter Drone designed by SF Technology.**

Sections 5.2.1 and 5.2.2. 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 Δ_{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 Δ_{ub} suggests that B&P has provided a better feasible solution, and a negative value of Δ_{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. However, the average computing time of B&P is 0.4 seconds, which is shorter than that of CPLEX, i.e., 38.7 seconds. Notably, CPLEX has explored a large number of B&B nodes, i.e., 165704.55 on average, whereas B&P only explores 4.8 nodes on average. For 15-customer instances, B&P can solve all the instances to optimality, consuming 0.7 seconds on average. However, CPLEX can only solve 3 instances to optimality, and the average optimality gap is 21.31%. When comparing the final upper bounds provided by the two methods, we find that CPLEX has found the optimal solution for 15 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 2E-VRP-D defined in this study and the other is the 2E-VRP-D with one drone on each truck. Results are given in Table 3.

Table 3 shows that for our 2E-VRP-D, B&P can solve 15 out of 20 instances to optimality for 25-customer instances. For the other five instances, i.e., *Cardiff_25_04*, *Cardiff_25_10*, *Cardiff_25_12*, *Cardiff_25_14*, and *Cardiff_25_19*, B&P can provide good solutions within the time limit—the optimality gaps are 3.42%, 1.65%, 3.08%, 6.40%, and 0.86%, respectively. For 35-customer instances,

Table 2 **Performance comparison between B&P and CPLEX on small-size instances.**

	B&P					CPLEX						
Instance	UB	LB	Gap	#Node	Time	UB	LB	Gap	#Node	Time	Δ_{ub}	Δ_{lb}
Cardiff_10_01	1223.6	1223.6	0.00	2	**0.9**	1223.6	1223.6	0.00	90067	31.3	0.0	0.0
Cardiff_10_02	1905.1	1905.1	0.00	1	**0.1**	1905.1	1905.1	0.00	42613	8.1	0.0	0.0
Cardiff_10_03	1412.7	1412.7	0.00	1	**0.1**	1412.7	1412.7	0.00	62541	16.8	0.0	0.0
Cardiff_10_04	1633.3	1633.3	0.00	22	**2.1**	1633.3	1633.3	0.00	122403	29.7	0.0	0.0
Cardiff_10_05	1682.9	1682.9	0.00	1	**0.0**	1682.9	1682.9	0.00	78258	18.9	0.0	0.0
Cardiff_10_06	1221.8	1221.8	0.00	1	**0.1**	1221.8	1221.8	0.00	86258	18.6	0.0	0.0
Cardiff_10_07	1931.8	1931.8	0.00	1	**0.0**	1931.8	1931.8	0.00	45836	7.8	0.0	0.0
Cardiff_10_08	1578.2	1578.2	0.00	37	**2.1**	1578.2	1578.2	0.00	61530	13.3	0.0	0.0
Cardiff_10_09	2552.0	2552.0	0.00	1	**0.0**	2552.0	2552.0	0.00	173411	24.1	0.0	0.0
Cardiff_10_10	2045.2	2045.2	0.00	1	**0.0**	2045.2	2045.2	0.00	280265	45.1	0.0	0.0
Cardiff_10_11	2095.3	2095.3	0.00	1	**0.0**	2095.3	2095.3	0.00	10603	3.4	0.0	0.0
Cardiff_10_12	1737.3	1737.3	0.00	19	**1.3**	1737.3	1737.3	0.00	727717	146.3	0.0	0.0
Cardiff_10_13	1485.9	1485.9	0.00	1	**0.0**	1485.9	1485.9	0.00	17607	8.0	0.0	0.0
Cardiff_10_14	1926.6	1926.6	0.00	1	**0.0**	1926.6	1926.6	0.00	15018	3.9	0.0	0.0
Cardiff_10_15	1814.3	1814.3	0.00	1	**0.0**	1814.3	1814.3	0.00	1120690	302.4	0.0	0.0
Cardiff_10_16	1140.1	1140.1	0.00	1	**0.1**	1140.1	1140.1	0.00	37366	10.5	0.0	0.0
Cardiff_10_17	1733.8	1733.8	0.00	1	**0.0**	1733.8	1733.8	0.00	20554	5.0	0.0	0.0
Cardiff_10_18	1960.9	1960.9	0.00	1	**0.0**	1960.9	1960.9	0.00	195234	54.1	0.0	0.0
Cardiff_10_19	1533.7	1533.7	0.00	1	**0.0**	1533.7	1533.7	0.00	73482	15.8	0.0	0.0
Cardiff_10_20	1366.4	1366.4	0.00	1	**0.1**	1366.4	1366.4	0.00	52638	11.7	0.0	0.0
Average	1699.0	1699.0	0.00	4.8	**0.4**	1699.0	1699.0	0.00	165704.55	38.7	0.0	0.0
Cardiff_15_01	1332.0	1332.0	0.00	1	**2.0**	1353.6	1158.2	14.43	340389	3600.0	1.6	-15.0
Cardiff_15_02	1970.6	1970.6	0.00	1	**0.3**	1970.6	1343.4	31.83	3623287	3600.0	0.0	-46.7
Cardiff_15_03	1971.4	1971.4	0.00	3	**0.6**	1997.3	1674.0	16.19	71354	3600.0	1.3	-17.8
Cardiff_15_04	2188.3	2188.3	0.00	2	**0.3**	2188.3	1799.7	17.76	2356896	2845.38*	0.0	-21.6
Cardiff_15_05	1831.3	1831.3	0.00	1	**0.1**	1831.3	1828.0	0.18	4425233	3600.0	0.0	-0.2
Cardiff_15_06	1695.2	1695.2	0.00	11	**2.7**	1695.2	1558.7	8.05	2125384	3600.0	0.0	-8.8
Cardiff_15_07	2050.7	2050.7	0.00	1	**0.2**	2050.7	1412.5	31.12	1972041	3600.0	0.0	-45.2
Cardiff_15_08	1689.6	1689.6	0.00	1	**0.2**	1689.6	1689.6	0.00	1946166	1373.9	0.0	0.0
Cardiff_15_09	2765.7	2765.7	0.00	1	**0.2**	2765.7	1577.7	42.95	5931427	3600.0	0.0	-75.3
Cardiff_15_10	2985.0	2985.0	0.00	17	**3.0**	2985.0	1447.2	51.52	11309109	3600.0	0.0	-106.3
Cardiff_15_11	2243.0	2243.0	0.00	1	**0.1**	2243.0	2243.0	0.00	1948711	1737.6	0.0	0.0
Cardiff_15_12	2155.8	2155.8	0.00	2	**0.9**	2166.4	1780.9	17.79	1698848	3600.0	0.5	-21.1
Cardiff_15_13	1606.5	1606.5	0.00	1	**0.2**	1606.5	1218.1	24.18	1266154	3600.0	0.0	-31.9
Cardiff_15_14	2369.9	2369.9	0.00	24	**2.1**	2373.2	1608.6	32.22	2062926	2318.91*	0.1	-47.3
Cardiff_15_15	1883.4	1883.4	0.00	1	**0.3**	1905.2	960.8	49.57	815672	2280.69*	1.1	-96.0
Cardiff_15_16	1756.9	1756.9	0.00	1	**0.5**	1756.9	1225.8	30.23	5036345	3600.0	0.0	-43.3
Cardiff_15_17	1858.2	1858.2	0.00	1	**0.1**	1858.2	1575.0	15.24	1681642	2261.13*	0.0	-18.0
Cardiff_15_18	2288.7	2288.7	0.00	2	**0.5**	2288.7	1476.0	35.51	4539072	3600.0	0.0	-55.1
Cardiff_15_19	1762.2	1762.2	0.00	1	**0.1**	1762.2	1762.2	0.00	1168490	995.7	0.0	0.0
Cardiff_15_20	1697.1	1697.1	0.00	1	**0.1**	1697.1	1569.9	7.50	2530498	3600.0	0.0	-8.1
Average	2005.1	2005.1	0.00	3.7	**0.7**	2009.2	1545.5	21.31	2842482.2	3030.7	0.2	-32.9

*: CPLEX ran out of memory.

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 0.09%. 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 2E-VRP-D 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

Table 3 Performance of B&P on medium-size instances.

	2E-VRP-D defined in this study					2E-VRP-D with one drone on each truck				
Instance	UB	LB	Gap	#Node	Time	UB	LB	Gap	#Node	Time
Cardiff_25_01	2233.9	2233.9	0.00	48	2741.1	2297.7	2297.7	0.00	6	374.8
Cardiff_25_02	1980.0	1980.0	0.00	9	7327.5	2005.2	2005.2	0.00	4	24.9
Cardiff_25_03	2389.7	2389.7	0.00	2	346.7	2396.3	2396.3	0.00	2	36.8
Cardiff_25_04	2801.8	2705.8	3.42	193	10800.0	2819.3	2819.3	0.00	1202	4504.9
Cardiff_25_05	2063.0	2063.0	0.00	1	51.1	2253.3	2253.3	0.00	136	482.5
Cardiff_25_06	2075.5	2075.5	0.00	1	12.2	2148.4	2148.4	0.00	10	35.7
Cardiff_25_07	2112.6	2112.6	0.00	2	55.6	2345.6	2345.6	0.00	30	119.3
Cardiff_25_08	2605.5	2605.5	0.00	1	48.9	2608.8	2608.8	0.00	1	8.7
Cardiff_25_09	2055.2	2055.2	0.00	2	174.7	2216.1	2216.1	0.00	80	203.3
Cardiff_25_10	2220.7	2183.9	1.65	32	10800.0	2327.4	2327.4	0.00	4	307.1
Cardiff_25_11	2051.1	2051.1	0.00	12	554.7	2286.5	2286.5	0.00	82	308.0
Cardiff_25_12	4783.1	4635.7	3.08	1489	10800.0	4797.9	4797.9	0.00	8934	10612.0
Cardiff_25_13	2248.6	2248.6	0.00	24	1973.0	2333.3	2333.3	0.00	40	230.8
Cardiff_25_14	2561.4	2397.6	6.40	108	10800.0	2561.4	2510.1	2.00	419	10800.0
Cardiff_25_15	2871.6	2871.6	0.00	67	1061.3	2882.3	2882.3	0.00	59	215.8
Cardiff_25_16	2503.3	2503.3	0.00	183	2037.1	2503.3	2503.3	0.00	25	101.1
Cardiff_25_17	2275.8	2275.8	0.00	1	414.3	2285.0	2285.0	0.00	1	9.3
Cardiff_25_18	2167.9	2167.9	0.00	34	185.5	2291.9	2291.9	0.00	118	340.0
Cardiff_25_19	2257.8	2238.3	0.86	415	10800.0	2276.8	2276.8	0.00	37	431.8
Cardiff_25_20	2816.6	2816.6	0.00	1	245.9	2977.8	2977.8	0.00	1	20.3
c101_35	1981.1	1981.1	0.00	194	3830.5	2086.1	2086.1	0.00	55	112.0
c102_35	1963.3	1961.4	0.09	278	10800.0	2057.5	2057.5	0.00	59	2292.3
c103_35	2267.9	1893.0	16.53	6	10800.0	2017.4	2017.4	0.00	190	3304.2
c104_35	2275.1	1887.1	17.05	2	10800.0	2009.6	2000.1	0.47	145	10800.0
c105_35	1965.1	1965.1	0.00	628	4822.2	2070.5	2070.5	0.00	69	326.1
c106_35	1971.9	1971.9	0.00	180	1423.2	2086.8	2086.8	0.00	68	310.3
c107_35	1946.0	1946.0	0.00	77	2596.4	2062.2	2062.2	0.00	45	210.6
c108_35	2274.5	1883.7	17.18	3	10800.0	1984.1	1984.1	0.00	1	36.4
c109_35	2275.0	1877.0	17.49	2	10800.0	1984.1	1984.1	0.00	1	96.6

Cardiff_25_14 and *c104_35*, the optimality gaps are 2.00% and 0.47%, respectively. Thus, we conclude that our B&P can solve most medium-size instances efficiently, especially for the 2E-VRP-D with one drone on each truck.

5.2.3. Performance of Tabu Search and Labeling Algorithm. To evaluate the TS algorithm for heuristic column generation and the exact labeling algorithm, we conduct experiments on instances with 10, 15, and 25 customers. We consider three cases to solve the pricing problem: only the TS is used, only the labeling algorithm is used (represented by *LA*), and both methods are applied (denoted as *TS+LA*). For the first case, the TS was run five times for each instance, and the average results are reported. In Tables 4 and 5, UB^* is the best upper bound (i.e., the best feasible solution) generated by the three methods, and Gap' is the percentage difference between UB^* and UB, computed as $Gap' = (UB - UB^*)/UB \times 100$. Since both LA and TS+LA solve all the small-size instances to optimality, we do not report their UBs in Table 4.

From Table 4, we observe that the B&P with the TS as the method for tackling the pricing problem can solve most 10-customer instances to optimality. However, it can only solve 3 out

Table 4 Performance of the tabu search and the labeling algorithm on 10- and 15-customer instances.

		TS			LA	TS+LA			TS			LA	TS+LA
Instance	UB*	UB	Gap′	Time	Time	Time	Instance	UB*	UB	Gap′	Time	Time	Time
Cardiff_10_01	1223.6	1245.7	1.77	0.4	0.4	0.9	Cardiff_15_01	1332.0	1419.8	6.19	0.4	1.9	2.0
Cardiff_10_02	1905.1	1905.8	0.03	0.1	0.0	0.1	Cardiff_15_02	1970.6	2017.0	2.30	0.1	0.6	0.3
Cardiff_10_03	1412.7	1412.7	0.00	0.1	0.1	0.1	Cardiff_15_03	1971.4	2071.3	4.82	0.1	4.7	0.6
Cardiff_10_04	1633.3	1661.8	1.71	1.2	1.3	2.1	Cardiff_15_04	2188.3	2828.8	22.64	0.0	0.4	0.3
Cardiff_10_05	1682.9	1699.5	0.97	0.0	0.0	0.0	Cardiff_15_05	1831.3	1877.9	2.48	0.1	0.2	0.1
Cardiff_10_06	1221.8	1221.8	0.00	0.1	0.0	0.1	Cardiff_15_06	1695.2	1705.2	0.59	2.5	2.3	2.7
Cardiff_10_07	1931.8	1931.8	0.00	0.1	0.0	0.0	Cardiff_15_07	2050.7	2070.7	0.97	0.1	0.6	0.2
Cardiff_10_08	1578.2	1596.5	1.15	1.2	2.4	2.1	Cardiff_15_08	1689.6	1706.7	1.00	0.1	0.3	0.2
Cardiff_10_09	2552.0	2552.0	0.00	0.0	0.0	0.0	Cardiff_15_09	2765.7	2888.9	4.26	0.1	0.1	0.2
Cardiff_10_10	2045.2	2045.2	0.00	0.0	0.1	0.0	Cardiff_15_10	2985.0	3027.7	1.41	0.7	1.2	3.0
Cardiff_10_11	2095.3	2095.3	0.00	0.0	0.0	0.0	Cardiff_15_11	2243.0	2253.9	0.49	0.1	0.1	0.1
Cardiff_10_12	1737.3	1754.4	0.98	0.1	0.9	1.3	Cardiff_15_12	2155.8	2189.0	1.52	0.0	4.6	0.9
Cardiff_10_13	1485.9	1636.4	9.20	0.0	0.0	0.0	Cardiff_15_13	1606.5	1620.5	0.86	0.1	0.2	0.2
Cardiff_10_14	1926.6	1950.1	1.20	0.0	0.0	0.0	Cardiff_15_14	2369.9	2369.9	0.00	1.5	2.6	2.1
Cardiff_10_15	1814.3	1814.3	0.00	0.0	0.0	0.0	Cardiff_15_15	1883.4	1915.7	1.69	0.1	1.4	0.3
Cardiff_10_16	1140.1	1180.5	3.43	0.1	0.1	0.1	Cardiff_15_16	1756.9	1790.2	1.86	0.7	3.2	0.5
Cardiff_10_17	1733.8	1733.8	0.00	0.0	0.0	0.0	Cardiff_15_17	1858.2	1858.2	0.00	0.1	0.2	0.1
Cardiff_10_18	1960.9	1960.9	0.00	0.0	0.0	0.0	Cardiff_15_18	2288.7	2318.0	1.26	0.6	0.7	0.5
Cardiff_10_19	1533.7	1533.7	0.00	0.0	0.0	0.0	Cardiff_15_19	1762.2	1762.2	0.00	0.1	0.1	0.1
Cardiff_10_20	1366.4	1461.2	6.48	0.0	0.0	0.1	Cardiff_15_20	1697.1	1741.1	2.53	0.0	0.1	0.1
Average	1699.0	1719.7	1.35	0.2	0.3	0.4	Average	2005.1	2071.6	2.84	0.4	1.3	0.7

Table 5 Performance of the tabu search and the labeling algorithm on 25-customer instances.

		TS			LA			TS+LA		
Instance	UB*	UB	Gap′	Time	UB	Gap′	Time	UB	Gap′	Time
Cardiff_25_01	2233.9	2540.4	12.07	2.0	2233.9	0.00	8154.0	2233.9	0.00	2741.1
Cardiff_25_02	1980.0	2188.8	9.54	8.1	1980.0	0.00	9731.3	1980.0	0.00	7327.5
Cardiff_25_03	2389.7	3079.3	22.39	0.2	2389.7	0.00	2133.8	2389.7	0.00	346.7
Cardiff_25_04	2801.8	3645.8	23.15	0.3	4752.8	41.05	10800.0	2801.8	0.00	10800.0
Cardiff_25_05	2063.0	2700.6	23.61	0.3	2063.0	0.00	280.9	2063.0	0.00	51.1
Cardiff_25_06	2075.5	2671.7	22.32	15.1	2075.5	0.00	61.4	2075.5	0.00	12.2
Cardiff_25_07	2112.6	2876.7	26.56	0.2	2112.6	0.00	198.5	2112.6	0.00	55.6
Cardiff_25_08	2605.5	3456.4	24.62	0.2	2605.5	0.00	248.4	2605.5	0.00	48.9
Cardiff_25_09	2055.2	2494.5	17.61	0.1	2055.2	0.00	1685.3	2055.2	0.00	174.7
Cardiff_25_10	2220.7	2878.6	22.86	0.2	3478.2	36.15	10800.0	2220.7	0.00	10800.0
Cardiff_25_11	2051.1	2766.3	25.85	0.1	2061.0	0.48	6815.4	2051.1	0.00	554.7
Cardiff_25_12	4783.1	6937.2	31.05	0.1	5035.1	5.00	10800.0	4783.1	0.00	10800.0
Cardiff_25_13	2248.6	2615.3	14.02	2.7	2248.6	0.00	4734.3	2248.6	0.00	1973.0
Cardiff_25_14	2561.4	3015.8	15.07	1.5	4857.7	47.27	10800.0	2561.4	0.00	10800.0
Cardiff_25_15	2871.6	4311.3	33.39	0.1	2871.6	0.00	138.6	2871.6	0.00	1061.3
Cardiff_25_16	2503.3	3540.6	29.30	0.2	2503.3	0.00	5824.9	2503.3	0.00	2037.1
Cardiff_25_17	2275.8	2645.8	13.98	0.2	2275.8	0.00	4136.4	2275.8	0.00	414.3
Cardiff_25_18	2167.9	2739.9	20.88	0.3	2167.9	0.00	1016.4	2167.9	0.00	185.5
Cardiff_25_19	2257.8	2682.1	15.82	1.0	4027.0	43.93	10800.0	2257.8	0.00	10800.0
Cardiff_25_20	2816.6	4050.8	30.47	0.1	2816.6	0.00	3194.1	2816.6	0.00	245.9
Average	2453.8	3191.9	21.73	1.6	2830.5	8.69	5117.7	2453.8	0.00	3561.5

of 20 instances with 15 customers to optimality, and the average gap between the UB^* and the UB is 2.84%. Regarding the computing time, the differences among the three methods are minor. From Table 5, we find that although the TS-based B&P consumes the least computing time, it generates poor solutions. Specifically, the generated upper bounds are much larger than the best

upper bounds, resulting in an average gap of 21.73%. When the exact labeling algorithm is used for solving the pricing problem, the corresponding B&P can solve most instances to optimality. However, the gaps between the UB^* and the UB are significant for four instances, i.e., 41.05%, 36.15%, 47.27%, and 43.93%, respectively. When the TS and the labeling algorithm are simultaneously employed, the corresponding B&P provides the best results, and the average computing time is also much shorter than the B&P with only the labeling algorithm, i.e., 3561.5 seconds against 5117.7 seconds. Thus, we conclude that the simultaneous application of the TS and the exact labeling algorithm can provide a good trade-off between solution quality and computing time.

5.3. Value of Drone Delivery and Drone Allocation

This section quantifies the benefits (i.e., the savings of time duration) of drone delivery and drone allocation decisions. We consider three cases: the VRP (i.e., no drone is involved in the delivery system), the VRPD with non-allocation decisions, and the VRPD with allocation decisions (i.e., our 2E-VRP-D). 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 optimally solved, and results are reported in Table 6. Indicators Gap_n^v and Gap_a^v are computed as $(D_v - D_n)/D_v \times 100$ and $(D_v - D_a)/D_v \times 100$, respectively. Parameter D_v is the duration of all routes in the optimal VRP solution. Parameters D_n and D_a are the durations under the non-allocation and allocation decisions, respectively. Thus, indicators Gap_n^v and Gap_a^v denote the duration savings obtained from using drones as delivery devices. The indicator Gap_a^n is computed as $(D_n - D_a)/D_n \times 100$, i.e., it quantifies the benefit obtained from allocation decisions. The last column in Table 6 reports the maximal value of Gap_a^n among all the instances in a set.

Table 6 Duration saving from using drones and drone allocation decisions.

			Total duration (average result)						
Instance	Γ	A_0	VRP	Non-allocation	Allocation	Gap_n^v	Gap_a^v	Gap_a^n	MaxGap
Cardiff_10	2	1	1747.56	1708.94	1699.04	2.21	2.78	0.58	5.91
Cardiff_15	2	1	2037.47	2028.96	2005.07	0.42	1.59	1.18	6.19
Cardiff_25	3	2	2407.32	2321.83	2296.68	3.55	4.60	1.08	5.49
Cardiff_25	3	1	2407.32	2388.77	2306.62	0.77	4.18	3.44	10.30

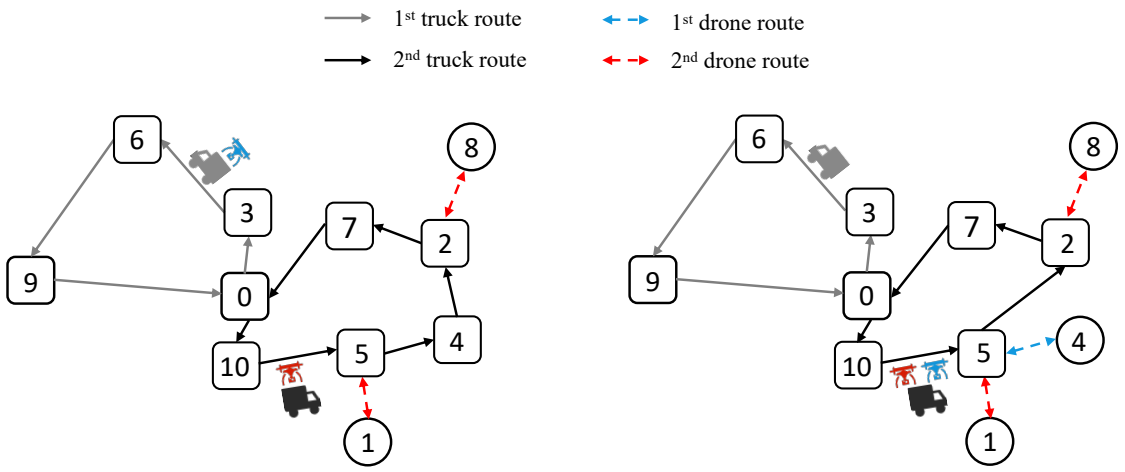
Columns 7 and 8 in Table 6 show that drones can help save the total duration. Specifically, compared with the VRP solution, the VRPD under non-allocation decisions can save the total duration from 0.42% to 3.55% for instances of different sizes on average. When the allocation decisions are considered, the average duration saving ranges from 1.59% to 4.60%. When comparing the solutions of the non-allocation and allocation decisions, the penultimate column demonstrates that

the allocation decisions can help save the total duration, from 0.58% to 1.18%. The maximal saving can reach 5.91%, 6.19%, and 5.49% for some 10-, 15-, and 25-customer instances, respectively. In addition, we notice that the values of Γ 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 for allocation decisions increases to 3.44%, and the maximal gap can reach 10.30%. This phenomenon suggests that our allocation decisions particularly benefit a delivery system with limited drones.

Table 7 **Average number of drones sent out from each satellite.**

Instance	#Drone	Instance	#Drone	Instance	#Drone
Cardiff_10_01	2.0	Cardiff_15_01	2.0	Cardiff_25_01	2.3
Cardiff_10_02	0.0	Cardiff_15_02	2.0	Cardiff_25_02	1.5
Cardiff_10_03	0.0	Cardiff_15_03	1.5	Cardiff_25_03	2.0
Cardiff_10_04	2.0	Cardiff_15_04	1.5	Cardiff_25_04	2.0
Cardiff_10_05	2.0	Cardiff_15_05	2.0	Cardiff_25_05	2.3
Cardiff_10_06	1.0	Cardiff_15_06	1.0	Cardiff_25_06	2.3
Cardiff_10_07	0.0	Cardiff_15_07	0.0	Cardiff_25_07	1.8
Cardiff_10_08	1.5	Cardiff_15_08	1.7	Cardiff_25_08	2.0
Cardiff_10_09	1.0	Cardiff_15_09	1.5	Cardiff_25_09	2.0
Cardiff_10_10	1.0	Cardiff_15_10	1.0	Cardiff_25_10	2.3
Cardiff_10_11	0.0	Cardiff_15_11	1.0	Cardiff_25_11	2.5
Cardiff_10_12	2.0	Cardiff_15_12	1.0	Cardiff_25_12	3.0
Cardiff_10_13	2.0	Cardiff_15_13	2.0	Cardiff_25_13	2.0
Cardiff_10_14	1.0	Cardiff_15_14	1.0	Cardiff_25_14	0.0
Cardiff_10_15	1.0	Cardiff_15_15	1.7	Cardiff_25_15	3.0
Cardiff_10_16	1.7	Cardiff_15_16	1.5	Cardiff_25_16	1.0
Cardiff_10_17	0.0	Cardiff_15_17	0.0	Cardiff_25_17	2.0
Cardiff_10_18	1.0	Cardiff_15_18	1.5	Cardiff_25_18	1.8
Cardiff_10_19	0.0	Cardiff_15_19	0.0	Cardiff_25_19	2.5
Cardiff_10_20	1.0	Cardiff_15_20	2.5	Cardiff_25_20	2.3
Average	1.0		1.3		2.0

To provide more details about the impacts of drone allocation decisions, Table 7 reports the average number of drones sent out from each satellite, denoted by *#Drone*. Note that satellites refer to the vehicle nodes where drones are launched. It shows that 1 drone is dispatched from each satellite on average for 10-customer instances. This number increases with the number of customers. For 25-customer instances, the value of #Drone reaches 2.0. Figure 6 gives an example to show the different routes generated under non-allocation and allocation decisions, where customers 2 and 5 are satellites. We find that each truck carries one drone when no drone allocation decision is involved (the left figure). However, the first drone is not utilized because customers in the first truck's route are quite far away from each other. When allocation decisions are involved, the second truck carries two drones, and no drone is allocated to the first truck. In this way, the



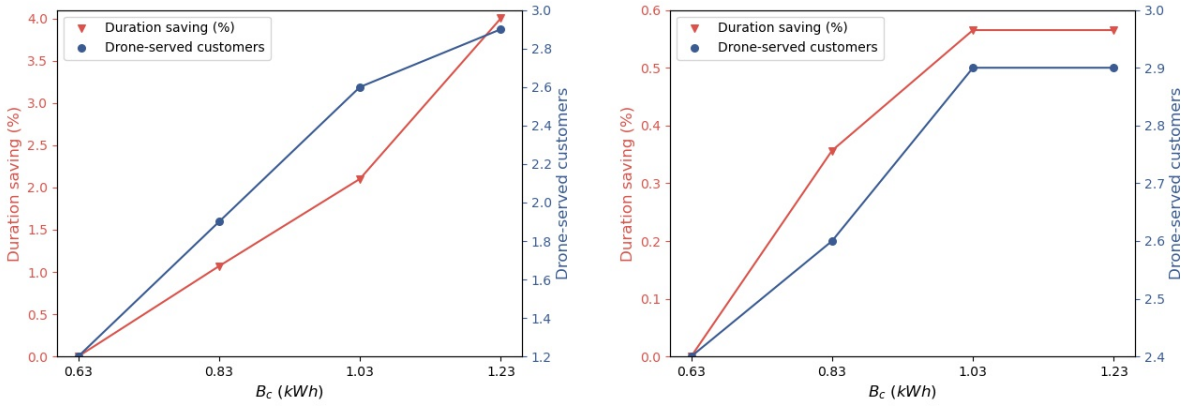
(a) Non-allocation (b) Allocation

Figure 6 **Solutions of instance Cardiff_10_08 with/without drone allocation.**

second truck can dispatch drones to serve customers 1 and 4 together when stopping at customer 5, leading to a 1.26% duration saving, i.e., 1578.2 against 1598.1.

5.4. Sensitivity Analyses

Since our 2E-VRP-D is a new variant of the VRPD, we conduct sensitivity analyses on key parameters, including battery energy capacity B_c, the maximum number of drones that a vehicle can carry Γ, and drone speed vel^d, to investigate the characteristics of this variant. Experiments are performed on 10- and 15-customer instances.



(a) 10-customer instances (b) 15-customer instances

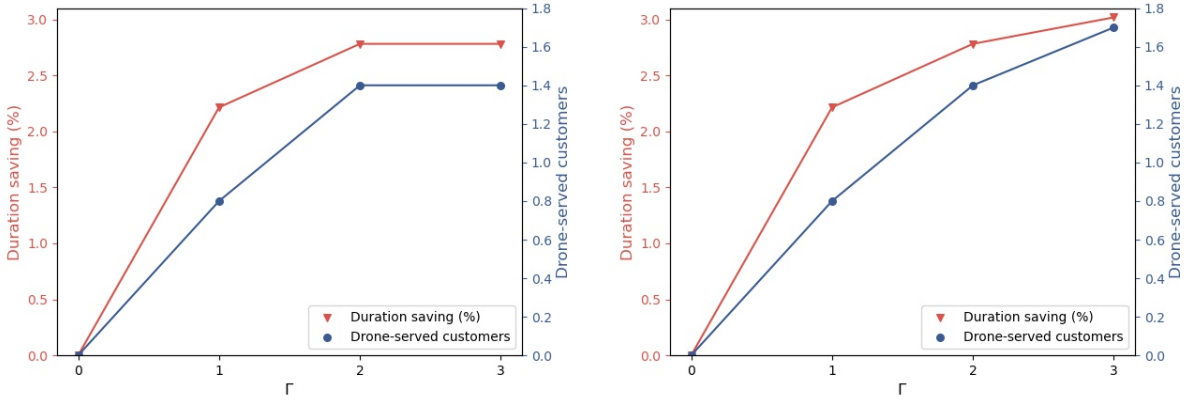
Figure 7 **The impact of battery energy capacity.**

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 0.63 to 1.23 with a step size of 0.2. Figure 7 plots the average results, where the duration saving is calculated with respect to the result under $B_c = 0.63\ kWh$. 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 $0.63\ kWh$ to $1.23\ kWh$, the time duration can be reduced by around 4.00% and 0.57% for 10- and 15-customer instances, respectively. We notice that the duration saving for 15-customer instances is smaller than that for 10-customer instances. To investigate if the number of customers has an impact on duration saving, i.e., as the number of customers increases, the system becomes less sensitive to the change in battery capacity; we conduct a similar experiment using the 25-customer instances by reducing the number of customers to 16, 12, and 8. Results show that the duration saving on these three instance sets are respectively 4.28%, 5.16%, and 6.55% when B_c changes from $0.63\ kWh$ to $1.23\ kWh$. One possible explanation for this phenomenon is that more customers are used as satellites when the number of customers is larger. That is, fewer customers are served by drones even though the flying radius of drones becomes larger (due to the increased energy capacity). This explanation can be supported by observing the lines denoting the drone-served customers in Figure 7. They show that when B_c increases from $0.63\ kWh$ to $1.23\ kWh$, the increased number of drone-served customers in 10-customer instances (i.e., 1.7) is more significant than that in 15-customer instances (i.e., 0.5). Correspondingly, the proportion of satellites in 10-customer instances is smaller. However, we note that large-size instances like 100-customer instances need to be examined before reaching the conclusion, which requires a more powerful exact algorithm. Thus, we leave it for future work.

Maximum Allowable Number of Drones on a Vehicle. When $\Gamma = 0$, the 2E-VRP-D reduces to a VRP with deadlines. Now we analyze the benefits of using a truck-drone system with different values of Γ. As drones and their support equipment also occupy vehicle capacity, we consider two different values for parameter Q, i.e., $Q = 170\ kg$ and $Q = 220\ kg$. The average results are reported in Figure 8, where the duration saving is calculated with respect to the case of $\Gamma = 0$. Figure 8 shows that the truck-drone system can help reduce the total duration as expected, compared to the truck-only system. We notice that when $Q = 170\ kg$, i.e., the vehicle capacity is relatively small, the comparison results in duration saving keep stable when Γ 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 $220\ kg$, more time can be saved when Γ varies from 2 to 3, particularly for the 15-customer instances.

Drone Speed. Now, we study the impact of drone speed vel^d. Intuitively, a higher drone speed will reduce the total duration; however, it also means higher energy consumption, influencing



(a) 10-customer instances with $Q = 170\ kg$

(b) 10-customer instances with $Q = 220\ kg$

(c) 15-customer instances with $Q = 170\ kg$

(d) 15-customer instances with $Q = 220\ kg$

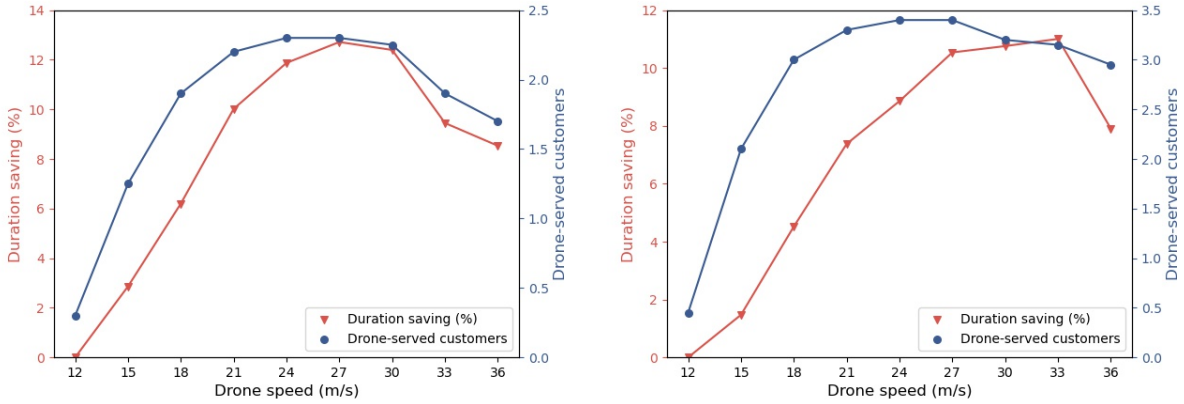
Figure 8 **The impact of the maximum allowable number of drones on a vehicle.**

drones' service region (Zeng et al. 2019). Thus, when drone speed is included in the energy function, we should decide an optimal speed to balance the duration reduction and the feasible service region. To study the impact of vel^d, we adopt the energy function in Zeng et al. (2019) and Dukkanci et al. (2021), written as

$$E(vel^d, d_{ij}) = \mu_1 \frac{d_{ij}}{vel^d} + \mu_2 d_{ij} vel^d + \mu_3 \frac{d_{ij}}{vel^{d2}} + \mu_4 d_{ij} (vel^d)^2,$$

where $\mu_i, i = 1, \ldots, 4$ are parameters, whose values are set as those in the aforementioned two studies. Since the weights of parcels and drones in our problem are larger, we set $B_c = \Theta/8 = 0.011\ kWh$, where Θ is a parameter similar to B_c in Dukkanci et al. (2021).

The relationship between the drone speed and the energy consumption per unit distance presents a "U" shape. Our preliminary tests show that the optimal drone speed minimizing the energy consumption is approximately 18 m/s. To show the full impact of vel^d, we conduct experiments by varying it from 12 m/s to 36 m/s. The average results are plotted in Figure 9, where the



(a) 10-customer instances

(b) 15-customer instances

Figure 9 The impact of drone speed.

duration saving is computed with respect to the case of $vel^d = 12\ m/s$. When $vel^d = 12\ m/s$, i.e., drone speed and vehicle speed are identical, only a minimal number of customers are served by drones. When vel^d increases from $vel^d = 12\ m/s$ to 27 m/s, more customers are served by drones; however, this number subsequently decreases when vel^d further increases. The duration saving also presents a similar trend. When vel^d is around 27 m/s, the duration saving reaches the maximum value, around 12.7%, for 10-customer instances. When vel^d is around 33 m/s, the time duration can be reduced by 11.0% for 15-customer instances. In general, increasing drone speed helps save drones' travel time, whereas it may lead to higher energy consumption, reducing drones' flying range or service region. Our results show that the resulting time saving is more significant than the impact on the service region when the drone speed falls within a certain range. However, the service region will be seriously decreased when the drone speed exceeds a particular value, limiting the applications of drones.

6. Conclusions

This paper introduces a new variant of the VRPD, which extends the classic truck-drone delivery problem to a two-echelon network and also considers 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 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 algorithm. To accelerate the CG procedure, a tabu search algorithm is first applied before the exact labeling algorithm to find routes with negative reduced costs. Extensive numerical tests based on two types of benchmark instances are conducted to evaluate the performance of the B&P and explore the impacts

of critical parameters. Results show that our B&P can solve most instances within 25 customers to optimality in a short time frame and some instances with 35 customers to optimality within a 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.

We consider the proposed model, algorithm, and numerical results to have the following practical implications. First, the 2E-VRP-D model provides decision-makers more flexibility to plan delivery routes for a truck-drone system, especially when the number of drones is limited, a typical phenomenon for most delivery companies. Second, although the developed B&P algorithm can only tackle 35-customer instances efficiently within an acceptable time, it can be used as a benchmark to evaluate heuristic or metaheuristic algorithms developed for solving large-size instances in future research. Third, the numerical results shed light on the benefits decision-makers can obtain by integrating drones into a delivery system, which may help them to rethink the configuration of their delivery systems. Moreover, results also suggest the marginal benefits they can gain by applying various technical strategies.

Future research could be conducted from three aspects: (1) In our energy function, we only consider the impact of payload or drone speed, other factors such as wind speed are neglected. Thus, more complex energy functions with multiple impact factors can be applied for drone energy calculation. One can also treat drone speeds as decision variables instead of given parameters to optimize the trade-off between service duration and service range as in Raj and Murray (2020). (2) We consider customers' service deadlines, a special case of the time windows. Thus, future research can consider a more general case, i.e., the time window constraints, and design tailored labeling algorithms. (3) Heuristic and metaheuristic algorithms can be developed to solve medium- or large-size instances more efficiently.

Acknowledgements

This work was supported by the National Key R&D Program of China [Grant 2018YFB1700600] and the National Natural Science Foundation of China [Grants 72101049, 71971090, 71821001, 72232001]. The authors are thankful to the associate editor and three anonymous referees for their helpful comments.

Declarations of Interest: None.

References

Agatz N, Bouman P, Schmidt M (2018) Optimization approaches for the traveling salesman problem with drone. *Transportation Science* 52(4):965–981.

Bakach I, Campbell AM, Ehmke JF (2021) A two-tier urban delivery network with robot-based deliveries. *Networks* 78(4):461–483.

Bakir I, Tiniç GÖ (2020) Optimizing drone-assisted last-mile deliveries: the vehicle routing problem with flexible drones. Preprint, submitted April 09, `http://www.optimization-online.org/DB_FILE/2020/04/7737.pdf`.

Bouman P, Agatz N, Schmidt M (2018) Dynamic programming approaches for the traveling salesman problem with drone. *Networks* 72(4):528–542.

Boysen N, Schwerdfeger S, Weidinger F (2018) Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research* 271(3):1085–1099.

Braekers K, Ramaekers K, Van Nieuwenhuyse I (2016) The vehicle routing problem: State of the art classification and review. *Computers & Industrial Engineering* 99:300–313.

Cavani S, Iori M, Roberti R (2021) Exact methods for the traveling salesman problem with multiple drones. *Transportation Research Part C: Emerging Technologies* 130:103280.

Chen C, Demir E, Huang Y (2021a) An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots. *European Journal of Operational Research* 294(3):1164–1180.

Chen C, Demir E, Huang Y, Qiu R (2021b) The adoption of self-driving delivery robots in last mile logistics. *Transportation Research Part E: Logistics and Transportation Review* 146:102214.

Cheng C, Adulyasak Y, Rousseau LM (2020) Drone routing with energy function: Formulation and exact algorithm. *Transportation Research Part B: Methodological* 139:364–387.

Chung SH, Sah B, Lee J (2020) Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research* 123:105004.

Dellaert N, Dashty Saridarq F, Van Woensel T, Crainic TG (2019) Branch-and-price–based algorithms for the two-echelon vehicle routing problem with time windows. *Transportation Science* 53(2):463–479.

Desaulniers G, Lessard F, Hadjar A (2008) Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows. *Transportation Science* 42(3):387–404.

Di Puglia Pugliese L, Guerriero F, Scutellá MG (2021a) The last-mile delivery process with trucks and drones under uncertain energy consumption. *Journal of Optimization Theory and Applications* 191(1):31–67.

Di Puglia Pugliese L, Macrina G, Guerriero F (2021b) Trucks and drones cooperation in the last-mile delivery process. *Networks* 78(4):371–399.

Dorling K, Heinrichs J, Messier GG, Magierowski S (2016) Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 47(1):70–85.

Dukkanci O, Kara BY, Bektaş T (2021) Minimizing energy and cost in range-limited drone deliveries with speed optimization. *Transportation Research Part C: Emerging Technologies* 125:102985.

El-Adle AM, Ghoniem A, Haouari M (2021) Parcel delivery by vehicle and drone. *Journal of the Operational Research Society* 72(2):398–416.

Glover F (1989) Tabu search—part I. *ORSA Journal on Computing* 1(3):190–206.

Glover F, Laguna M (1998) Tabu search. *Handbook of combinatorial optimization*, 2093–2229 (Springer).

Kang M, Lee C (2021) An exact algorithm for heterogeneous drone-truck routing problem. *Transportation Science* 55(5):1088–1112.

Kitjacharoenchai P, Min BC, Lee S (2020) Two echelon vehicle routing problem with drones in last mile delivery. *International Journal of Production Economics* 225:107598.

Kuo R, Lu SH, Lai PY, Mara STW (2022) Vehicle routing problem with drones considering time windows. *Expert Systems with Applications* 191:116264.

Leishman GJ (2006) *Principles of helicopter aerodynamics with CD extra* (Cambridge University Press).

Li H, Chen J, Wang F, Bai M (2021) Ground-vehicle and unmanned-aerial-vehicle routing problems from two-echelon scheme perspective: A review. *European Journal of Operational Research* 294(3):1078–1095.

Li H, Chen J, Wang F, Zhao Y (2022) Truck and drone routing problem with synchronization on arcs. *Naval Research Logistics (NRL)* 69(6):884–901.

Li H, Wang H, Chen J, Bai M (2020) Two-echelon vehicle routing problem with time windows and mobile satellites. *Transportation Research Part B: Methodological* 138:179–201.

Macrina G, Pugliese LDP, Guerriero F, Laporte G (2020) Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies* 120:102762.

Marques G, Sadykov R, Deschamps JC, Dupas R (2020) An improved branch-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Computers & Operations Research* 114:104833.

Moshref-Javadi M, Lee S, Winkenbach M (2020) Design and evaluation of a multi-trip delivery model with truck and drones. *Transportation Research Part E: Logistics and Transportation Review* 136:101887.

Murray CC, Chu AG (2015) The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies* 54:86–109.

Otto A, Agatz N, Campbell J, Golden B, Pesch E (2018) Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks* 72(4):411–458.

Poikonen S, Campbell JF (2021) Future directions in drone routing research. *Networks* 77(1):116–126.

Poikonen S, Golden B, Wasil EA (2019) A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing* 31(2):335–346.

Qin H, Su X, Ren T, Luo Z (2021) A review on the electric vehicle routing problems: Variants and algorithms. *Frontiers of Engineering Management* 8(3):370–389.

Raj R, Murray C (2020) The multiple flying sidekicks traveling salesman problem with variable drone speeds. *Transportation Research Part C: Emerging Technologies* 120:102813.

Righini G, Salani M (2006) Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization* 3(3):255–273.

Roberti R, Ruthmair M (2021) Exact methods for the traveling salesman problem with drone. *Transportation Science* 55(2):315–335.

Salama M, Srinivas S (2020) Joint optimization of customer location clustering and drone-based routing for last-mile deliveries. *Transportation Research Part C: Emerging Technologies* 114:620–642.

Santos FA, da Cunha AS, Mateus GR (2013) Branch-and-price algorithms for the two-echelon capacitated vehicle routing problem. *Optimization Letters* 7(7):1537–1547.

Santos FA, Mateus GR, da Cunha AS (2015) A branch-and-cut-and-price algorithm for the two-echelon capacitated vehicle routing problem. *Transportation Science* 49(2):355–368.

SF Technology (n.d.) SF Technology-Ark Octocoper Drone. Accessed September 4, 2022, `https://www.sf-tech.com.cn/en/product/ark-octocoper-drone`.

Sluijk N, Florio AM, Kinable J, Dellaert N, Van Woensel T (2023) Two-echelon vehicle routing problems: A literature review. *European Journal of Operational Research* 304(3):865–886.

Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35(2):254–265.

Stolaroff JK, Samaras C, O'Neill ER, Lubers A, Mitchell AS, Ceperley D (2018) Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery. *Nature Communications* 9(1):1–13.

Stonor C (2021) New amazon patent for delivery van-controlled drones. Accessed September 04, 2021, `https://www.urbanairmobilitynews.com/express-delivery/new-amazon-patent-for-delivery-van-controlled-drones/`.

Stroh AM, Erera AL, Toriello A (2022) Tactical design of same-day delivery systems. *Management Science* 68(5):3444–3463.

Tamke F, Buscher U (2021) A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological* 144:174–203.

Ulmer MW, Thomas BW (2018) Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks* 72(4):475–505.

Unmanned Airspace (2017) Mercedes Zurich drone delivery programme reaches 100 flights. Accessed August 30, 2022, `https://www.unmannedairspace.info/uncategorized/mercedes-zurich-drone-delivery-programme-reaches-100-flights/`.

Vásquez SA, Angulo G, Klapp MA (2021) An exact solution method for the tsp with drone based on decomposition. *Computers & Operations Research* 127:105127.

Vu L, Vu DM, Hà MH, Nguyen VP (2022) The two-echelon routing problem with truck and drones. *International Transactions in Operational Research* 29(5):2968–2994.

Wang Z, Sheu JB (2019) Vehicle routing problem with drones. *Transportation Research Part B: Methodological* 122:350–364.

Yu S, Puchinger J, Sun S (2020) Two-echelon urban deliveries using autonomous vehicles. *Transportation Research Part E: Logistics and Transportation Review* 141:102018.

Yu S, Puchinger J, Sun S (2022) Van-based robot hybrid pickup and delivery routing problem. *European Journal of Operational Research* 298(3):894–914.

Zeng Y, Xu J, Zhang R (2019) Energy minimization for wireless communication with rotary-wing uav. *IEEE Transactions on Wireless Communications* 18(4):2329–2345.

Zhen L, Gao J, Tan Z, Wang S, Baldacci R (2022) Branch-price-and-cut for trucks and drones cooperative delivery. *IISE Transactions* 1–29.

Appendix A: Proofs of Statements

This sections provides the proofs of statements in the main paper.

A.1. Proof of Proposition 1

Consider two labels L_f^1 and L_f^2 that satisfy conditions (38)–(43). We first prove $\Xi(L_f^2) \subseteq \Xi(L_f^1)$. For any partial route $p^b \in \Xi(L_f^2)$, let p_1 be one of the routes obtained by combining $p(L_f^1)$ with p^b. Condition (40) guarantees that p_1 satisfies the vehicle capacity constraint. Let $S(p)$ be the set of nodes visited along p. Conditions (41)–(42) guarantee that $S(p^b) \subseteq \left(\Omega_1(L_f^2) \cup \Omega_2(L_f^2)\right) \subseteq \left(\Omega_1(L_f^1) \cup \Omega_2(L_f^1)\right)$, so $S(p(L_f^1)) \cap S(p^b) = \emptyset$ and p_1 is an elementary route. Condition (38) guarantee that the drone nodes visited from $\sigma(L_f^1)$ satisfy the battery energy constraint for route p_1. Finally, according to condition (39), $p(L_f^1)$ can depart earlier at all drone routes and the vehicle route than $p(L_f^2)$. Due to $p^b \in \Xi(L_f^2)$, there must exist a route $p_2 \in \mathcal{R}(p(L_f^2) \oplus p^b)$. If we connect $p(L_f^1)$ and p^b with the same order as p_2 to obtain p_1, then p_1 must satisfy customers' deadlines. To summarize, for any $p^b \in \Xi(L_f^2)$, we can always find a corresponding route $p_1 \in \mathcal{R}(p(L_f^1) \oplus p^b)$. Thus, for all $p^b \in \Xi(L_f^2)$, the relation $p^b \in \Xi(L_f^1)$ holds, and then we have $\Xi(L_f^2) \subseteq \Xi(L_f^1)$.

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

$$
\begin{aligned}
&\bar{C}(p_1) - \bar{C}(p_2) \\
&= \max\left\{\max_{i=1,\ldots,\bar{k}_d}\left\{\pi^i(L_f^1) + \rho^i(p^b)\right\}, \pi^0(L_f^1)\right\} + t^v_{\sigma(L_f^1)\sigma(p^b)} + C_2(p^b) - C(L_f^1) - C_1(p^b) \\
&\quad - \max\left\{\max_{i=1,\ldots,\bar{k}_d}\left\{\pi^i(L_f^2) + \rho^i(p^b)\right\}, \pi^0(L_f^2)\right\} - t^v_{\sigma(L_f^2)\sigma(p^b)} - C_2(p^b) + C(L_f^2) + C_1(p^b) \\
&= \max\left\{\max_{i=1,\ldots,\bar{k}_d}\left\{\pi^i(L_f^1) + \rho^i(p^b) - C(L_f^1)\right\}, \pi^0(L_f^1) - C(L_f^1)\right\} \\
&\quad - \max\left\{\max_{i=1,\ldots,\bar{k}_d}\left\{\pi^i(L_f^2) + \rho^i(p^b) - C(L_f^2)\right\}, \pi^0(L_f^2) - C(L_f^2)\right\} \\
&\leq \max\left\{\max_{i=1,\ldots,\bar{k}_d}\left\{\pi^i(L_f^1) + \rho^i(p^b) - C(L_f^1)\right\} - \max_{i=1,\ldots,\bar{k}_d}\left\{\pi^i(L_f^2) + \rho^i(p^b) - C(L_f^2)\right\},\right. \\
&\qquad \left.\pi^0(L_f^1) - C(L_f^1) - \pi^0(L_f^2) + C(L_f^2)\right\} \\
&\leq \max\left\{\max_{i=1,\ldots,\bar{k}_d}\left\{\pi^i(L_f^1) - C(L_f^1) - \pi^i(L_f^2) + C(L_f^2)\right\}, \pi^0(L_f^1) - C(L_f^1) - \pi^0(L_f^2) + C(L_f^2)\right\} \\
&= \max_{i=0,\ldots,\bar{k}_d}\left\{\pi^i(L_f^1) - C(L_f^1) - \pi^i(L_f^2) + C(L_f^2)\right\} \\
&\leq 0.
\end{aligned}
$$

The third and fourth inequalities are obtained based on Theorem A1 in the following. As there may exist multiple combinations of $\pi^i(L_f)$ satisfying constraints (39) and (43), we only need to find one combination to guarantee the relations established.

Theorem A1 *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 A1: 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

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

A.2. 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 $\max_{i=1,\ldots,n}\{a_i - b_{k_i}\} < \max_{i=1,\ldots,n}\{a_i - b_i\}$, which is denoted as *order B*. Let $i' = \text{argmax}_{i=1,\ldots,n}\{a_i - b_i\}$ and $j' = \text{argmax}_{j=1,\ldots,n}\{a_j - b_{k_j}\}$. According to the assumption $a_{j'} - b_{k_{j'}} < a_{i'} - b_{i'}$, in *order B* there must exist an index $k_{i'} \in \{i'+1, i'+2, \ldots, n\}$, otherwise the relation $a_{j'} - b_{k_{j'}} \geq a_{i'} - b_{k_{i'}}$ 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 $a_{i^*} - b_{k_{i^*}} \geq a_{i'} - b_{i'} > a_{j'} - b_{k_{j'}}$. However, only $n - i'$ elements are in this set, which makes it impossible to construct a bijection.

A.3. Proof of Proposition 2

Before proving Proposition 2, we give the following definitions:

Definition A1 ***(Final optimal)*** *Suppose we have two labels L_f^1 and L_f^2 satisfying (i) $\sigma(L_f^1) = \sigma(L_f^2) = \sigma(L_f)$, (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 $\max_{i=0,\ldots,\bar{k}_d}\{\pi^i(L_f^1)\} \leq \max_{i=0,\ldots,\bar{k}_d}\{\pi^i(L_f^2)\}$, we call L_f^1 a final optimal label. It means that for all drone nodes to be served from $\sigma(L_f)$, label L_f^1 must be one of the best labels when we are extending a vehicle node.*

Definition A2 ***(The same type of final optimal)*** *For label L_f^1 and L_f^2, if (i) $\sigma(L_f^1) = \sigma(L_f^2) = \sigma(L_f)$, (ii) the partial routes before reaching $\sigma(L_f)$ are the 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 A3 ***(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.*

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 $\bar{\pi}^i(L_f)$ be the completion time when removing its last node. Let $i^* = \text{argmax}_{i=1,\dots,\bar{k}_d}\{\bar{\pi}^i(L_f)\}$ and $j^* = \text{argmin}_{j=1,\dots,\bar{k}_d}\{\pi^j(L_f)\}$. If there exists a route j satisfying $\bar{\pi}^{i^*}(L_f) > \pi^{j^*}(L_f)$, 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^*. These movement operations only advance the service start times of removed nodes, thus L_f^* must be a feasible label. As both $\pi^{i^*}(L_f^*)$ and $\pi^{j^*}(L_f^*)$ are smaller than $\pi^i(L_f)$, L_f^* 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 Proposition 2. Let L_f be a final optimal* label. We continue to use the definitions of $\bar{\pi}^i(L_f)$ and i^* above, and remove the last drone node in route i^* to obtain label L'_f. Due to $\pi^{i^*}(L'_f) = \min_{i=1,\dots,\bar{k}_d}\{\pi^i(L'_f)\}$, we can obtain L_f from L'_f using Proposition 2. We then prove L'_f is a final optimal* label. If removing the last node in route k ($k \neq i^*$), then we have $\bar{\pi}^k(L'_f) = \bar{\pi}^k(L_f) \leq \bar{\pi}^{i^*}(L_f) = \pi^{i^*}(L'_f)$. And for route i^*, the relation $\bar{\pi}^{i^*}(L'_f) < \pi^{i^*}(L'_f) = \bar{\pi}^{i^*}(L_f) \leq \pi^k(L_f) = \pi^k(L'_f)$ holds, so L'_f is a final optimal* label.

A.4. Proof of Proposition 3

We only give the proof of $\bar{C}(p_1) \leq \bar{C}(p_2)$. The proof of $\Xi(L_b^2) \subseteq \Xi(L_b^1)$ can refer to the proof of Proposition 1. Let $\Xi(L_b)$ 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 $\pi^i(p^f)$ and $C(p^f)$ have the same meanings as $\pi^i(L_f)$ and $C(L_f)$, respectively. Then we have

$$
\begin{aligned}
&\bar{C}(p_1) - \bar{C}(p_2) \\
&= \max\left\{\max_{i=1,\dots,\bar{k}_d}\left\{\pi^i(p^f) + \rho^i(L_b^1)\right\}, \pi^0(p^f)\right\} + C_2(L_b^1) - C(p^f) - C_1(L_b^1) \\
&\quad - \max\left\{\max_{i=1,\dots,\bar{k}_d}\left\{\pi^i(p^f) + \rho^i(L_b^2)\right\}, \pi^0(p^f)\right\} - C_2(L_b^2) + C(p^f) + C_1(L_b^2) \\
&= \max\left\{\max_{i=1,\dots,\bar{k}_d}\left\{\pi^i(p^f) + \rho^i(L_b^1) + C_2(L_b^1) - C_1(L_b^1)\right\}, \pi^0(p^f) + C_2(L_b^1) - C_1(L_b^1)\right\} \\
&\quad - \max\left\{\max_{i=1,\dots,\bar{k}_d}\left\{\pi^i(p^f) + \rho^i(L_b^2) + C_2(L_b^2) - C_1(L_b^2)\right\}, \pi^0(p^f) + C_2(L_b^2) - C_1(L_b^2)\right\} \\
&\leq \max\left\{\max_{i=1,\dots,\bar{k}_d}\left\{\pi^i(p^f) + \rho^i(L_b^1) + C_2(L_b^1) - C_1(L_b^1)\right\} - \max_{i=1,\dots,\bar{k}_d}\left\{\pi^i(p^f) + \rho^i(L_b^2) + C_2(L_b^2) - C_1(L_b^2)\right\},\right. \\
&\quad \left. C_2(L_b^1) - C_1(L_b^1) - C_2(L_b^2) + C_1(L_b^2)\right\} \\
&\leq \max\left\{\max_{i=1,\dots,\bar{k}_d}\left\{\rho^i(L_b^1) + C_2(L_b^1) - C_1(L_b^1) - \rho^i(L_b^2) - C_2(L_b^2) + C_1(L_b^2)\right\},\right. \\
&\qquad \left. C_2(L_b^1) - C_1(L_b^1) - C_2(L_b^2) + C_1(L_b^2)\right\} \\
&= \max_{i=0,\dots,\bar{k}_d}\left\{\rho^i(L_b^1) + C_2(L_b^1) - C_1(L_b^1) - \rho^i(L_b^2) - C_2(L_b^2) + C_1(L_b^2)\right\} \\
&\leq 0.
\end{aligned}
$$

A.5. 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 $\pi^i(L_f) \leq \pi^j(L_b)$, and sort the set in non-descending order concerning the value of $\rho^j(L_b)$. As $\pi^1(L_f) \geq \pi^2(L_f) \geq ... \geq \pi^{\bar{k}_d}(L_f)$, 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 1, 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 by Algorithm 1, 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 1 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 $\pi^{i^*}(L_f) + \rho^{j^*}(L_b)$. 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 $\pi^{i^*}(L_f) + \rho^{j^*}(L_b)$. Therefore, $i^* - 1$ forward routes should be paired with $i^* - 2$ backward routes, which is impossible.