CIRRELT

CIRRELT-2024-34

A Dynamic Drone Routing Problem with Uncertain Demand and Energy Consumption

Patrizia Beraldi
Gulherme O. Chagas
Leandro C. Coelho
Demetrio Laganà

October 2024

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2024-006

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél : 1-418-656-2073
Télécopie : 1-418-656-2624

A Dynamic Drone Routing Problem with Uncertain Demand and Energy Consumption

Patrizia Beraldi[1], Gulherme O. Chagas[2,3], Leandro C. Coelho[2,3,4*], Demetrio Laganà[1,2]

1. Department of Mechanical, Energy and Management Engineering (DIMEG), University of Calabria, Via P. Bucci, Cubo 41C, 87036 - Arcavacata di Rende (CS), Italy
2. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)
3. Department of Operations and Decision Systems, Université Laval, Québec, Canada
4. Canada Research Chair in Integrated Logistics, Université Laval, Québec, Canada

Abstract. This work addresses a drone route problem with a homogeneous fleet and with same-day deliveries in a dynamic and uncertain environment. We model this problem as a Markov Decision Process to capture the stochastic nature of requests' demands and the uncertainty in energy consumption due to varying payloads and weather conditions. To tackle this problem, we propose an approximate dynamic algorithm that integrates routing planning, drone usage, and battery management. The approach employs chance constraints to ensure that drone trips are completed safely, considering energy uncertainties and preventing premature returns to the depot. The proposed approach features a cost function approximation policy that accounts for a restricted number of trips to be assigned to drones. This ensures that the drones are ready at the depot to fulfill new requests that may arise during the day. Extensive computational experiments in 300 instances validate our method's effectiveness, demonstrating its superiority over a myopic strategy and highlighting its potential for practical applications.

Keywords: Combinatorial optimization, drone routing, Markov decision process, cost function approximation, uncertain demand, uncertain energy consumption.

Acknowledgements. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) [grant number 2019-00094]. The work of Demetrio Laganà is partly supported by Ministero delle Imprese e del Made in Italy, Fondo per la Crescita Sostenibile - Accordi per l'innovazione di cui al D.M. 31 dicembre 2021 e D.D. 18 marzo 2022, project F/310044/01-04/X56 -- Crawford "advanCed seRvices and netWorks FOr aiR Drone transport".

---

* Corresponding author: leandro.coelho@fsa.ulaval.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2024

1. Introduction

Drone delivery can redefine the shipping market as drones offer a promising alternative to traditional delivery methods, especially in congested urban areas or locations that are difficult to reach by conventional vehicles. Moreover, they can contribute to sustainable development as electric-powered drones produce fewer emissions than traditional delivery vehicles (Garg et al., 2023). Large distributors like Amazon have already tried these innovative ways to deliver goods. These services have also emerged within the grocery and restaurant domain, e.g., Uber Eats Drone Delivery and KFC Drone fast food delivery focusing on delivering food within tight deadlines and time windows. Aside from commercial use, drones can deliver relief items in humanitarian logistics and refill medicines in healthcare projects.

Ongoing improvements in drone technology, such as increased payload capacity, extended range, and improved safety features, continue to expand the potential of drone delivery, posing new challenges in optimizing logistic operations. In this paper, we address the problem of optimizing a daily delivery service by exploiting a fleet of identical drones under demand and energy consumption uncertainty. This entails designing a system that dynamically creates trips and optimizes their assignment to drones, considering the evolving delivery requirements throughout the day.

In optimizing drone operations, batteries represent the most critical resource. Most medium-sized civilian drones are powered by lithium-ion batteries that can guarantee a limited airborne time. Thus, drones should make multiple trips with frequent stops at the depot for battery swaps, i.e., replacing the discharged battery with a fully charged one before performing another trip. Properly addressing battery usage and fully exploiting their range ensures an efficient system design. In our proposed approach, we build trips serving multiple requests, each delivering a small package to a customer location. In the rest of the paper, we use the terms *request*, *delivery*, *customer*, *delivery request* or *customer request* interchangeably.

The number of deliveries on a trip depends on the total payload limited by the drone capacity and energy consumption. Our approach assumes that consumption is a function of the payload and flight time. Considering load-dependent consumption is essential for accurate energy estimates since heavier loads require more energy. In addition, drones are more sensitive to weather conditions than traditional vehicles. For example, wind can influence the drone's speed

and, thus, the flight time and range. To account for uncertainty, we assume that the energy consumption is random, and we employ the paradigm of chance constraints to build safe trips, avoiding routing the drones back to the depot prematurely and delaying the delivery service.

We model our problem setting as a Markov decision process (MDP) to represent the dynamic stochastic nature of the request arrival. Moreover, since we consider time-sensitive requests, we allow customers to be served after their soft deadline. The objective of our problem is to define routing plans that maximize the number of customers served while minimizing the total expected cost, which is the sum of the expected routing cost and the cost of the expected lateness to make deliveries. Finally, dynamic drone dispatching is carried out at the depot to ensure equal battery usage.

Our work contributes to the literature on drone delivery problems, defining a decision approach that accounts for the dynamic stochastic process of request arrival, the uncertainty affecting energy consumption, and the deterioration of batteries. More specifically, our paper makes the following contributions to existing literature:

- From a modeling perspective, we propose a decision approach where routing decisions also account for ground service operations, i.e., charging and assignment of the batteries to drones;
- In a novel approach, we address two significant issues typically dealt with separately in the existing literature. In particular, we propose an MDP model accounting for the dynamic and stochastic nature of the arrival process and for uncertainty in energy consumption by applying the paradigm of chance constraints;
- Methodologically, we present a tailored cost function approximation (CFA) policy to facilitate decision-making. The policy is defined based on decomposing decisions into simple and sequential actions that are taken by solving mathematical formulations in which the optimal solution of one provides the input for the next. A calibrated threshold on the maximum number of trips assigned to each drone at each decision epoch is defined. This represents a means of accounting for future requests when a decision is taken;
- The paper provides extensive computational experiments on instances derived from the literature. The results demonstrate the effectiveness of our approach versus myopic strate-

gies.

In our contribution, we focus on a same-day delivery problem, and we assume that the delivery requests are not known in advance, but they arrive randomly during the day. Each request is associated with a (soft) deadline, limiting the time between its realization and the service time. This temporal dimension, reflecting the time sensitivity in the fulfillment of the customer requests, makes our framework general and applicable in other contexts, such as in emergency medical services, where requests with tight due dates require immediate attention and have a higher priority in the route planning phase.

The paper is organized as follows. In Section 2, we position our work concerning the literature. Section 3 describes the problem, whereas Section 4 presents the mathematical modeling of the problem as an MDP. The solution approach of our cost function approximation is described in Section 5, where its steps and actions are detailed in Section 5.3. In Section 6, we outline the data sets and present the results of computational tests of our solution approach. Finally, Section 7 provides some conclusions and discusses future work.

2. Literature Review

Delivery problems involving drones are typically classified into pure and combined truck-drone problems. In the first case, drones represent the sole vehicles operated by the service provider, whereas in the second case, drones cooperate with ground vehicles to complete the service to customers. Routing problems of the second class have been extensively studied since the introduction of the first problem combining a truck and a drone by Murray and Chu (2015). Later on, different variants of the problem have been investigated. We refer interested readers to Macrina et al. (2020), where the authors provide a drone problem taxonomy and classification according to the different cooperation models.

The drone-only problem, often framed as a variant of the vehicle routing problem (VRP), has received comparatively less attention. Dorling et al. (2017) proposed a multi-trip vehicle routing formulation in which a single drone at a depot executes multiple trips to satisfy the delivery requests. A notable contribution was examining the relationship between the drone's energy consumption and the total payload, including the battery weight. Addressing this relationship led to the formulation of a challenging mixed-integer programming problem solved by

a simulated annealing heuristic algorithm.

Coelho et al. (2017) studied a multi-objective routing problem where charging stations are used to extend the limited range of drones. The proposed problem has seven different objective functions, one of which is the minimization of the energy required by the drone batteries. However, the energy required is estimated using a simple function that depends on the drone's speed instead of a realistic consumption model.

Cheng et al. (2020a) extended the multi-trip formulation proposed in Dorling et al. (2017) by modeling the energy consumption as a nonlinear function of payload and travel distance. The problem is solved by a specialized branch-and-cut algorithm where logical cuts and subgradient cuts are introduced to tackle the nonlinear (convex) energy consumption function. We also refer the reader to Zhang et al. (2021) for a review, classification, and assessment of the main drone energy consumption models proposed in the scientific literature.

A few papers consider uncertainty in energy consumption in the drone delivery problems. Among them, we cite Pugliese et al. (2021), where the authors acknowledged the importance of considering uncertainty in energy consumption and model the drone and truck problem with the robust optimization paradigm. In Cheng et al. (2020b), the authors analyze the influence of weather conditions (mainly wind) in drone flight and propose a two-period data-driven adaptive distributionally robust approach where wind observation data are used to improve scheduling decisions. Specifically, the scheduling decisions for the fleet of drones are made in the morning, with the provision for different delivery schedules in the afternoon that adapt to updated weather information available by midday. Unlike our approach, the delivery requests are assumed to be known in advance.

In our contribution, we explicitly deal with uncertainty in energy consumption. As in Dorling et al. (2017), we assume that the energy consumption can be expressed as a function of the travel time and the load carried out by a drone. Uncertainty in the weather conditions affects travel time and, as a consequence, energy consumption. Moreover, considering load-dependent energy consumption makes the delivery problem even more challenging. As packages are delivered, the drone gets lighter, and energy consumption changes. Thus, accounting for load variation becomes essential for accurately estimating energy consumption. This also impacts the assignment of the delivery requests to the available drones to reduce the overall energy consumption during the delivery process. Unlike the contributions mentioned above, we employ

the chance constraint paradigm (Ruszczyński and Shapiro, 2003) to build trips that satisfy the energy requirement with a high probability level, thus avoiding drones being routed back to the depot before completing the assigned trip.

Another stream of literature related to our problem is the stochastic dynamic VRP (SD-VRP). Unlike static (deterministic) problems, where all information is assumed to be known at the time of decision-making, in a dynamic setting, routing actions are taken under incomplete information. The new logistic business models prioritize instant gratification and real-time fulfillment and are the main drivers of this paradigm shift. Applications range from ride-hailing services (Gao et al., 2024) to restaurant delivery (Ulmer et al., 2021), as well as emergency repair or health care services (van Steenbergen et al., 2023). Uncertainty is typically related to the arrival of new requests (see, e.g., Ulmer et al. (2019) and Zhang and Woensel (2023)) and changes in the fleet size, for example, by the inclusion of crowdsourced drivers and their behavior (Ulmer and Savelsbergh, 2020).

SDVRPs are generally modeled as MDPs, where a sequence of routing decisions is defined in reaction and anticipation of newly revealed stochastic information. Interested readers are referred to the recent review of Soeffker et al. (2022), where the authors analyze different contributions on SDVRP in the light of prescriptive analytics, focusing on integrating information models into decision models via computational methods. Also related to the problem considered in this work are the contributions of Chen et al. (2022, 2023), where the authors analyze a same-day delivery problem with stochastic customer requests. In these studies, the fleet comprises conventional vehicles and drones.

Another work related to our study is that of Ulmer and Thomas (2018), where the authors investigate the impact of using a heterogeneous fleet for delivery service in a dynamic routing problem for the first time. Uncertainty is related to customer requests being revealed throughout the day. Indeed, until the request is made, no temporal and spatial information is available. When a new request occurs, the service provider needs to decide whether the new request is accepted or not. If service is offered, it must be conducted before a deadline, within a predefined time after the request is placed, and the provider should further decide if the delivery must be performed by a vehicle or a drone. Modifications should be made to the planned trips to accommodate new requests and maximize the expected number of deliveries per day.

Similar to these contributions, this work assumes dynamic uncertainty in customer requests,

but we only consider a fleet of drones that may serve more than one customer per trip. More importantly, we deal with ground services operations that affect the definition of routing plans. Moreover, uncertainty in the energy consumption affecting the routing trip design is explicitly accounted for by integrating probabilistic constraints. Finally, the problem is tackled using an algorithm that iteratively and dynamically plans both the routes and drone deployment.

3. Problem description

This section formally describes the drone routing problem accounting for ground service operations. We focus on the same-day delivery problem and assume that a delivery company randomly receives requests during the day. The working day is discretized and represented by the time horizon $\mathcal{T} = \{0, 1, \ldots, T\}$. Each request r is associated with a tuple of information: the delivery location (i_r), the corresponding load (q_r), and a soft deadline ($l_r \in \mathcal{T}$) representing a time within which the request should be fulfilled after it is placed. In addition, a service time (η), assumed to be the same for all deliveries, is also considered.

Delivery requests come from an area that can be reached by drones based at a depot, also referred to as a charging station. The service area is represented by a complete undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where the set $\mathcal{N}$ includes the depot, identified by node i_0, and the vertices associated with the customer locations $\mathcal{N}' = \{i_1, i_2, \ldots, i_m\}$. The set of edges $\mathcal{A}$ includes all possible links between pairs of vertices, and each edge $\{i_r, i_p\}$ is associated with a travel distance (cost) denoted as $d_{i_r i_p}$, where the triangle inequality holds.

Delivery service is provided by a fleet of homogeneous drones identified by the set $\Delta = \{\delta_1, \delta_2, \ldots, \delta_n\}$, each with the same capacity Q and self-weight v, including the weight of both the drone itself and its battery. The batteries have limited energy capacity, thus allowing for limited airborne time. This implies that drones must repeatedly return to the depot to swap the battery, i.e., replacing the discharged battery with a fully charged one, before loading parcels for the next deliveries. A limited number of batteries is assumed to be available at the depot. These batteries and those equipped in the drones compose the set of batteries $\mathcal{B}$, where $|\mathcal{B}| = b$. Unlike other contributions, e.g., Dorling et al. (2017), we do not assume that the delivery company has enough fully charged batteries to meet the drone's energy needs for the day. Swapped-out batteries are charged at the depot while drones operate and are swapped into other drones in need once fully charged. We denote by ρ the average time needed to perform the swapping

operation and to equip drones. The recharging time depends on the residual energy of each used battery.

It is trivial to observe that accounting for charging operations makes the drone routing problem more challenging, as it requires the coordination of both the routing and the ground service phases. Batteries can typically withstand only a few hundred charge-discharge cycles in their lifetime (French, 2021), making them a consumable component in the system. When planning delivery operations, it is therefore important to use batteries uniformly and to ensure a balanced usage between them. In the proposed approach, the assignment of a battery to a drone is performed in such a way as to guarantee a balanced usage, ensuring that all batteries age equally. We aim to define trips that serve multiple requests and are compatible with the drone's carrying capacity and limited energy autonomy. In particular, the energy $e_{i_r i_p}$ a drone requires to fly between locations i_r and i_p is a function of the required power, which in turn depends on the payload $q_{i_r i_p}$ carried along the edge, in addition to the self-weight v, and on the travel time expressed as the ratio between the distance $d_{i_r i_p}$ and the observed drone speed $\xi_{i_r i_p}$. As in Dorling et al. (2017), the value of the $e_{i_r i_p}$ is calculated as:

$$e_{i_r i_p}(q_{i_r i_p}, \xi_{i_r i_p}) = \frac{d_{i_r i_p}}{\xi_{i_r i_p}} \cdot \left((v + q_{i_r i_p})^{\frac{3}{2}} \sqrt{\frac{\mathtt{g}^3}{2\mathtt{pzm}}} \right), \tag{1}$$

where $\mathtt{g}$ denotes the gravity acceleration, $\mathtt{p}$ is the air density, $\mathtt{z}$ is the area of the drone's spinning blade of each of its $\mathtt{m}$ rotors. Since the time unit used in this work for setting the drone speed is minutes, the required energy is measured in kW/min. Looking at Equation (1), it is easy to recognize that the energy consumption $e_{i_r i_p}$ is uncertain, as its speed $\xi_{i_r i_p}$ is affected by the weather (wind). In the following, we assume that this consumption is a random variable following a Normal distribution. To streamline the notation, we denote it as $\widetilde{e}_{i_r i_p}(q_{i_r i_p})$, omitting the dependence on drone speed, which will be handled in Section 5.3.

The problem tackled in this paper, referred to as the Drone Routing Problem with Uncertain Demand and Energy Consumption (DRPUDEC), aims to optimize ground and flight operations by ensuring a balanced use of the available batteries and accounting for random customer requests and random energy consumption in the route design. In particular, we propose an uncertainty-aware approach based on the MDP framework and the Approximate Dynamic Programming (ADP) methodology to deal with uncertain customer demands that may occur

throughout the day, and on the paradigm of chance constraints to ensure the construction of safe trips, i.e., trips that can be completed with a high probability under unfavorable conditions that may occur, avoiding that drones might be routed prematurely back to the depot. Routing plans are created to maximize the number of customers served while minimizing the expected total cost composed of (a) the routing costs measured in terms of the total expected distance traveled by drones and (b) the lateness cost due to the expected accumulated lateness as the approach allows to serve customer's requests after their soft deadlines. An overview of the notation used in the description is reported in Table 1.

Table 1: Notation summary.

Problem data	
b	number of batteries
m	number of requests (customers), where $r = \{1, \dots, m\}$
n	number of drones
$\mathcal{N}'$	set of customers nodes (locations), i.e., $\mathcal{N}' = \{i_1, i_2, \dots, i_m\}$
$\mathcal{G}$	graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N} = \mathcal{N}' \cup \{i_0\}$ is the set of customers and the depot and $\mathcal{A}$ is the set of edges
$d_{i_r i_p}$	distance between locations $i_r, i_p \in \mathcal{N}$
$\mathcal{T}$	time horizon, where $\mathcal{T} = \{0, 1, \dots, T\}$
$\mathcal{B}$	set of homogeneous batteries such that $\lvert\mathcal{B}\rvert = b$ and $\beta \in \mathcal{B}$
Δ	set of homogeneous drones such that $\lvert\Delta\rvert = n$ and $\delta \in \Delta$
Ψ	time interval between two consecutive epochs
$\bar{l}$	time range such that customers whose deadline l_r lies within $t_k + \bar{l}$ are considered urgent
μ^d, μ^l	monetary cost per unit of distance and lateness, respectively
Battery data	
ρ, e'_β	battery swapping time and recharge ratio, respectively
ω_k^β	number of times, up to epoch k, battery β was swapped
E_{min}, E_{max}	battery minimum and maximum energy level, respectively
Drone data	
$\mathtt{m}, \mathtt{z}, \mathtt{p}$	number of rotors in a drone, area of the drone's spinning blade, and fluid density of the air, respectively
$v, Q, \bar{\xi}$	drone's weight, capacity, and average speed, respectively
$\xi_{i_r i_p}$	drone's realized speed between locations $i_r \in \mathcal{N}$ and $i_p \in \mathcal{N}$
$\widetilde{e}_{i_r i_p}(q_{i_r i_p})$	energy consumption of a drone carrying $q_{i_r i_p}$ and flying between locations $i_r \in \mathcal{N}$ and $i_p \in \mathcal{N}$
M	maximum number of trips that can be assigned to a drone
Trip data	
$c(\gamma), \tau(\gamma), \psi(\gamma)$	total cost, length, and incurred lateness of a trip γ, respectively
$\mathcal{P}_\gamma$	sequence of request nodes visited by trip γ
$\widetilde{CE}_\gamma$	required energy to perform a trip γ
Request data	
η	delivery service time
i_r, q_r, l_r	request r location, demand, and soft deadline, respectively
State data and variables	
$\mathcal{K}$	set of decision epochs, where $k \in \mathcal{K}$
t_k	time of epoch k, where $t_k \in \mathcal{T}$
S_k, S_k^x	pre-decision and post-decision state, respectively
$\tau(S_k, x)$	total traveled distances incurred by action x when system in state S_k at epoch k
$\psi(S_k, x)$	lateness incurred by action x when system in state S_k at epoch k
$c(S_k, x)$	cost incurred by action x when in state S_k at epoch k, where $c(S_k, x) = \mu^d \cdot \tau(S_k, x) + \mu^l \cdot \psi(S_k, x)$
$\Delta_k, \mathcal{B}_k$	state of each drone and battery, respectively, at epoch k
$\Delta_k^x, \mathcal{B}_k^x$	state of each drone and battery, respectively, at epoch k due to the execution of action x
$\mathcal{X}(S_k)$	set of feasible actions in S_k, $x \in \mathcal{X}(S_k)$
$\mathcal{D}_k$	set of outstanding requests at epoch k
$\mathcal{D}_k^x$	set of outstanding requests due to the execution of action x at epoch k
$\mathcal{U}_k$	set newly arrived requests at epoch k
Γ_k	set of trips generated by Algorithm 3 at epoch k, where $\gamma \in \Gamma_k$ is a trip
$a_\delta(i_r)$	arrival time of drone $\delta \in \Delta$ at location i_r
θ_k^δ	route plan for drone $\delta \in \Delta$ at period k

4. An MDP formulation

To deal with the dynamic stochastic nature of the problem, we model the DRPUDEC as an MDP over a finite, discrete-time horizon, and we deal with uncertainty in energy consumption by adopting the chance constraint paradigm (Ruszczyński and Shapiro, 2003). We denote the set of decision epochs as $\mathcal{K}$ and by t_k the time associated with epoch $k \in \mathcal{K}$. We assume that decision epochs occur at regular intervals Ψ. In the following, we describe the main components of the MDP approach: the states (Section 4.1) and the MDP dynamics (Section 4.2). The methodology falls in the ADP and it is based on a cost function approximation (Sections 5 and 5.3).

4.1. States

At each epoch k, the system occupies a pre-decision state S_k, which contains all the relevant information to undertake routing and ground service decisions. Specifically, a state S_k comprises the current time of day t_k, the state of every request, the state of each drone, and the state of each battery at the charging station. Requests known at epoch k are partitioned into two subsets denoted as $\mathcal{D}_k$ and $\mathcal{U}_k$, respectively. Requests in $\mathcal{D}_k$ are called *outstanding* and are deliveries that already appeared in the system (before t_k) but not yet served because assigned to trips for which the loading of all the corresponding parcels scheduled in the trip to the drone have not been started at the charging station, whereas requests in $\mathcal{U}_k$ represent the *new customers* appearing in epoch k.

We denote by Δ_k the set of information describing the state of each drone at epoch k. Specifically, for each drone $\delta \in \Delta_k$ we know the assigned route plan θ_k^δ, which consists of a trip schedule $\left\{\gamma_k^\delta(1), \gamma_k^\delta(2), \ldots, \gamma_k^\delta(w)\right\}$, where $w \leq M$ is the last trip index assigned to the drone and M is the maximum number of trips that can be assigned to a drone. In the following, for the sake of simplicity, we shall omit symbols k and δ and the trip index if they are clear from the context. During a trip, a drone δ can visit several locations before returning to the depot to perform service tasks. We represent a generic trip γ as a sequence of nodes $\gamma = \{i_0, i_1, \ldots, i_r, \ldots, i_p, i_{p+1}\}$ starting and ending at the depot $i_0 = i_{p+1}$. For each node i_r in the trip, we also define the arrival time $a_\delta(i_r)$. Since the weather conditions influence the flight time in our approach, we shall consider the corresponding expected value of each trip's total time, cost, and lateness. Note that a deadline of $l_r \in \mathcal{T}$ is associated with each delivery. Lateness in service is computed as

$[a_\delta(i_r) - l_r]^+ = \max\{0, a_\delta(i_r) - l_r\}$. Once a parcel has been delivered, a drone can execute the remaining deliveries in the trip. We note that the departure time from a node can be defined as the sum of the arrival time and the service time η, which is assumed to be constant for all customers. We can thus associate with each trip a total expected lateness $\psi(\gamma)$. The total expected cost of a trip γ is given by

$$c(\gamma) = \mu^d \cdot \tau(\gamma) + \mu^l \cdot \psi(\gamma), \tag{2}$$

where $\tau(\gamma)$ is the total distance traveled by the drone when performing trip γ, and μ^d and μ^l are the coefficients representing the monetary cost per unit of distance and lateness, respectively.

When a drone δ returns to the depot, it is assigned a fully charged battery. We denote by ρ the service time associated with the battery swapping and loading of new parcels. If no fully charged battery is available, i.e., all batteries are being recharged, the drone has to wait for a battery to be available and assigned to it before leaving the depot for another trip. We denote by $\mathcal{B}_k$ the set of information describing the state of each battery at epoch k. Each battery is assigned a parameter ω_k^β, which indicates the number of times the battery has been swapped from the beginning of the time horizon until epoch k. This latter element guarantees a balanced utilization when assigning a battery to a drone if several fully recharged batteries can be assigned.

Formally, we describe a pre-decision state as $S_k = (t_k, \mathcal{D}_k, \mathcal{U}_k, \Delta_k, \mathcal{B}_k)$. In the initial state S_0, all drones are available at the depot with empty planned trips, batteries are fully charged, and ω_0^β are set to 0 for every battery $\beta \in \mathcal{B}_0$.

4.2. Dynamics of the DRPUDEC

At a decision epoch k, the decision maker observes the pre-decision state S_k and selects an action $x \in \mathcal{X}(S_k)$, where $\mathcal{X}(S_k)$ is the set of feasible actions corresponding to the pre-decision state S_k. An action x consists of two components: (i) it specifies the battery-drone assignment, and (ii) it updates the routing plan of the drones through the assignment and routing of newly revealed requests $r \in \mathcal{U}_k$ and the possible reassignment and re-sequencing of outstanding requests $r \in \mathcal{D}_k$. If the number of fully recharged batteries is lower than the number of available drones, a partial assignment is made to have some drones wait at the depot. During trip construction, the following requirements must be met:

a) the energy level must remain equal to or greater than a safety margin $E_{\min}$ to prevent the drone from returning to the depot prematurely. Since the energy consumption is uncertain, this condition is ensured by the chance constraint paradigm, which requires that the energy consumption to visit all the locations along a given trip is greater than $E_{\min}$, with probability α;

b) the requests served in a trip are selected following two rules: minimizing the distance traveled and reducing the lateness associated with serving requests beyond their stipulated deadline;

c) drones must return to the depot no later than the end of the day, denoted by T.

Executing an action x in a pre-decision state S_k entails the application of a decision rule, resulting in an immediate cost, denoted by $c(S_k, x)$, which is a contribution from two components: the total distance traveled by all drones ($\tau(S_k, x)$) and the total lateness in all trips ($\psi(S_k, x)$). Here, the cost $c(S_k, x)$ is computed similarly as in Equation (2). A sequence of decision rules defines a policy, denoted by $\pi = (X_0^\pi(S_0), X_1^\pi(S_1), \dots X_T^\pi(S_T))$, and Π denotes the set of all Markovian deterministic policies. The optimal policy is defined as the one that minimizes the total expected cost, given the initial state S_0:

$$\min_{\pi \in \Pi} \mathbb{E}\left[\sum_{k=1}^{T} c(S_k, X_k^\pi(S_k)) | S_0\right].$$

By using the Bellman equation, the cost structure can be recursively restated as:

$$V(S_k) = \min_{x \in \mathcal{X}(S_k)} \{c(S_k, x) + \mathbb{E}\left[V(S_{k+1}) | S_k^x\right]\} \tag{3}$$

for $k = 1, \dots, T-1$. At the end of the MDP, we have the total cost c, which is the sum of each component k, $c(S_k, x)$.

Algorithm 1 outlines the dynamics of the MDP. Line 1 initializes the initial state S_0, where all drones are available at the depot, all batteries are fully charged, and the first customers are revealed. Line 2 sets the initial epoch k and the total cost c to zero. Then, in line 4, the pre-decision state S_k transitions to post-decision S_k^x by taking action x. This step is detailed in Algorithm 2. Besides the post-decision S_k^x, the procedure of line 4 also returns the drones'

routing and the deliveries' lateness costs incurred in the deterministic transition to the post-decision state S_k^x. These values are added to the total cost.

Following the generation of S_k^x, the procedure entails the implementation of the stochastic transition to the next pre-decision state S_{k+1} at the epoch $k+1$, as shown in lines 6–11, which are detailed in Section 5.2. The steps of the while loop of lines 3–12 are repeated until the end of the time horizon is reached, and the total cost is returned.

Algorithm 1: Dynamics of the Markov decision process for the DRPUDEC.

input : Set of drones Δ; set of batteries $\mathcal{B}$.
output : Total cost c.

1 $S_0 = (t_0, \mathcal{D}_0, \mathcal{U}_0, \Delta_0, \mathcal{B}_0)$ `// setup initial state` S_0
2 $k \leftarrow 0,\ c \leftarrow 0$
3 **while** $t_k < T$ **do**
  `/* deterministic transition to post-decision state` S_k^x `from` S_k `*/`
4 $S_k^x = (t_k, \mathcal{D}_k^x, \emptyset, \Delta_k^x, \mathcal{B}_k^x), c(S_k, x) \leftarrow process\_decision\_state(S_k)$ `// call Algorithm 2`
5 $c \leftarrow c + c(S_k, x)$ `// update total cost`
  `/* stochastic transition to the pre-decision state` S_{k+1} `from` S_k^x `*/`
6 $\mathcal{U}_{k+1} = \{r_{\texttt{new}} = (i_{r_{\texttt{new}}}, q_{r_{\texttt{new}}}, l_{r_{\texttt{new}}})\}$ `// newly revealed requests`
7 $\mathcal{D}_{k+1} \leftarrow \mathcal{D}_k^x$ `// update outstanding customers set`
8 $\Delta_{k+1} \leftarrow \Delta_k^x$ `// update drones' states`
9 $\mathcal{B}_{k+1} \leftarrow \mathcal{B}_k^x$ `// update batteries' states`
10 $t_{k+1} \leftarrow t_k + \Psi$ `// update next time period`
11 $k \leftarrow k + 1$ `// advance to the next epoch`
12 **end**
13 **return** c

5. Solution Methodology

Identifying an optimal policy is challenging due to the three curses of dimensionality affecting the DRPUDEC: (*i*) the state space is multi-dimensional and can grow exponentially; (*ii*) the action space comprises an exponential number of options for assigning and sequencing delivery requests, and (*iii*) the outcome space is vast due to the uncertainty in the arrival of the requests and energy consumption. To overcome these challenges, one of four common strategies is typically adopted: lookahead approximation or rollout algorithms (LA or RA), value function approximation (VFA), policy function approximation (PFA), and cost function approximation (CFA) (Soeffker et al., 2022). LAs or RAs may require significant online computation to assess the future impact of an action. This may entail incorporating information about future realizations of requests in a predictive manner, particularly when a routing decision must be made,

as in the DRPUDEC. VFAs approximate the second term of the Bellman equation (3), but are limited by dimensionality, which can become quite large in routing problems with multiple drones. PFAs are best suited when it is possible to design a policy that captures the structure of the problem to provide decisions. This is a very problem-specific policy, which in the case of the DRPUDEC means designing a pre-defined policy that decides about the routing, which is also impractical or inefficient.

Since the DRPUDEC is affected by uncertainties regarding independent random variables (such as requests and energy consumption of drones), defining a policy based on a state-independent rule is challenging. CFAs modify a deterministic optimization formulation's objective and/or constraints to better account for future uncertainty. This method can be easily applied to the DRPUDEC where multiple optimization models are considered in sequence to build a policy that mitigates the inherent limitations of rolling horizon optimization, namely its tendency towards a rigid and inflexible decision-making process. In fact, in the DRPUDEC, it is crucial to ensure that decisions account for the opportunity to have a sufficient number of drones either at the charging station or flying back to it at crucial times. This way, they can be routed to deliver new requests revealed at each epoch. This aim can be achieved by defining the maximum number of trips that can be assigned to a single drone, allowing for more informed decision-making. In the proposed approach, the feasible regions of the optimization problems presented in Sections 5.3.3 and 5.3.4 are manipulated by the specified maximum number of trips. These problems are solved to optimality in sequence at each state of the system, as detailed in the following sections.

We observe that the computational resources for solving the current but manipulated decision model instances of the optimization problems described below are very limited compared to Ψ. This motivates using an *optimization problem-based CFA policy* for the DRPUDEC. More precisely, in line 3 of Algorithm 2, we specify that drones cannot be assigned more than M trips. This algorithm establishes a CFA policy known as $\pi^{CFA}(M)$ based on the particular value of M. In general, let $\Pi^{CFA} = \left\{\pi^{CFA}(M) : M \in \left\{1, \ldots, \overline{M}\right\}\right\}$ denote the set comprising all CFA policies for $M \in \left\{1, \ldots, \overline{M}\right\}$, where $\overline{M}$ is set large enough so that a fleet of n drones in the depot can service all outstanding and new requests in each epoch.

Let M^* be the value of $M \in \left\{1, \ldots, \overline{M}\right\}$ for which the minimum routing and lateness cost is obtained. When $M \neq M^*$, many drones may still be busy flying at the beginning of the next

epoch, which can lead to an accumulation of delivery lateness for new requests. As evidenced by the computational analysis presented in Section 6.1.1, the best value of M is a function of Ψ. A myopic policy $\pi^{CFA}(M = +\infty)$ is the one based on an unbounded number of trips that can be assigned and executed by a drone to serve all the outstanding and newly arrived requests in each state S_k. The trip schedule assigned to each drone is executed even if some trips of the schedule start in the next epoch. This myopic policy fails to consider the possibility of a limited number of scheduled trips being executed by each drone. Consequently, there is a risk of insufficient drones being available in the depot to fly in the next epoch, which could result in a large lateness in serving new requests in the pre-decision state S_{k+1}.

5.1. Deterministic transition to post-decision state

Algorithm 2 shows the deterministic transition to the post-decision state S_k^x from the pre-decision state S_k, regarding the design of the CFA policy as outlined in Section 5. Line 2 focuses on the trip-building step, which employs chance constraints to account for uncertainty in energy consumption. In this step, explained in Section 5.3.1, a backward heuristic is used to build trips, greedily adding nodes by non-decreasing distance or time to deadline. The backward heuristic generates the set Γ_k of trips that are not dominated, which is used as input for the procedure of line 3 that selects up to M trips per drone from this set.

In the procedure *select_trips*, we initially solve a set packing-based problem that seeks to identify the set $\Gamma'_k \subseteq \Gamma_k$ that serves the largest number of requests, with the highest priority given to those that are urgent, defined as those for which the soft deadline is near to the current epoch time. The solution of this model consists of at most M trips assigned to a drone. A second model is then solved, this time using the set Γ_k specifying which customers must be served. This model aims to select at most M trips per drone that avoid overlapping customer visits while minimizing the total routing cost. Further details regarding the algorithm in line 3 are provided in Section 5.3.2.

Solving the previous model allows us to find the set Γ_k^* of least routing cost drone trips, whereby the maximum number of customers are visited. This set can be partitioned into n trip subsets, each representing a potential assignment of M trips to a drone. When a trip schedule is assigned to a drone, additional lateness in serving customers on a trip can be accumulated due to the sub-optimal order in which the trips are executed in each schedule. Therefore, an optimal

Algorithm 2: Process decision state (*process_decision_state*).

input : Pre-decision state $S_k = (t_k, \mathcal{D}_k, \mathcal{U}_k, \Delta_k, \mathcal{B}_k)$.
output : Post-decision state $S_k^x = (t_k^x, \mathcal{D}_k^x, \emptyset, \Delta_k^x, \mathcal{B}_k^x)$; cost $c(S_k, x)$.

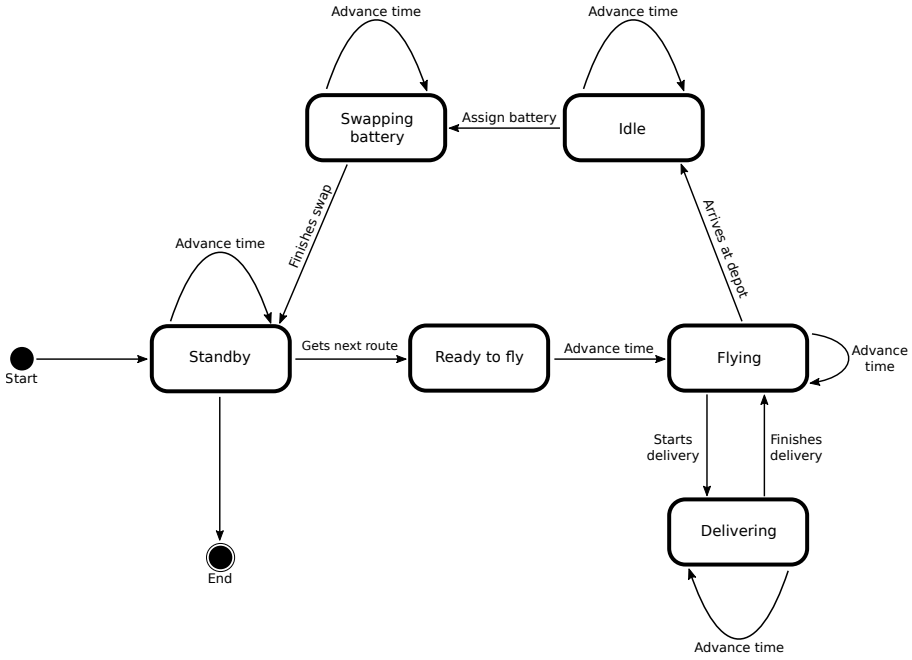
1 $\tau(S_k, x) \leftarrow 0,\ \psi(S_k, x) \leftarrow 0$
2 $\Gamma_k \leftarrow build\_trips(\mathcal{D}_k \cup \mathcal{U}_k)$ `// build set` Γ_k `by Algorithm 3`
3 $\Gamma_k^* \leftarrow select\_trips(\mathcal{D}_k \cup \mathcal{U}_k, \Gamma_k, |\Delta|, M)$ `// find set` Γ_k^* `by Algorithm 5`
4 $\left\{\theta_k^\delta : \delta \in \Delta_k\right\} \leftarrow assign\_trips(\Gamma_k^*, \Delta_k)$ `// assign trips to drones by Algorithm 6`
5 **foreach** $\delta \in \Delta_k$ **do**
  `/* execute state machine, depicted by Figure 1, of drone` δ `for` Ψ `units of time. During this execution, update sets` $\Delta_k^x, \mathcal{B}_k^x, \mathcal{D}_k^x$ `and update costs` $\tau(S_k, x)$ `and` $\psi(S_k, x)$ `accordingly. Also, update` ω_k^β `for every battery` β `swapped */`
6 $execute\_drone\_state\_machine(\delta, \theta_k^\delta, \mathcal{B}_k^x, \tau(S_k, x), \psi(S_k, x))$
  `/* after executing drone` δ `state machine for` Ψ `units of time, its future trips are canceled in preparation for the next epoch. Then, all customers contained on these not performed trips are returned to set` $\mathcal{D}_k^x$ `*/`
7 $\mathcal{D}_k^x \leftarrow \mathcal{D}_k^x \cup cancel\_scheduled\_trips(\delta)$
8 **end**
9 **return** $S_k^x, \mu^d \cdot \tau(S_k, x) + \mu^l \cdot \psi(S_k, x)$

permutation of the trips assigned to a drone must be identified to minimize customer service lateness. This is achieved by exhaustively considering all possible trip orders and selecting the schedule with the least lateness. The least lateness trip schedules are then used in a third model, which is tasked with assigning each schedule to a drone in a manner that minimizes overall lateness while accounting for the state of the drones at the current epoch time. These steps are executed on line 4, where θ_k^δ is the trip schedule assigned to drone $\delta \in \Delta_k$. We observe that scenarios with fewer trip schedules than drones are possible. Therefore, assignment solutions where one or more drones are not used are also valid. Further details of the algorithm referenced in line 4 are provided in Section 5.3.5.

After the assignment step, each drone $\delta \in \Delta_k$ executes its assigned route plan θ_k^δ, if any, following the algorithm outlined in line 6. Figure 1 depicts the dynamic of the DRPUDEC, describing a trip schedule that could be executed by a drone in any potential scenario. This is referred to as the drone finite-state machine. It operates for each $\delta \in \Delta_k$ over a duration of Ψ units of time, during which it updates sets $\Delta_k^x, \mathcal{B}_k^x, \mathcal{D}_k^x$. Additionally, it aggregates the cost $c(S_k, x)$ incurred by the trips executed by drone δ within the Ψ time span. Upon completing its operation after Ψ units of time, the drone idles at its current state and position within the

state machine and resumes from the same state and position in the subsequent epoch $k + 1$.

Figure 1: Drone execution finite-state machine.



Following the execution of the drone δ state machine for a duration of Ψ, any remaining future trips that start after the current time period $t_k + \Psi$ are canceled in preparation for the next epoch as outlined in line 7 of Algorithm 2. We observe that trips currently underway by a drone and the ones starting before $t_k + \Psi$ are not affected. Then, all customers associated with the unexecuted trips are returned to the set $\mathcal{D}_k^x$. After executing all drones for Ψ units of time, the post-decision state S_k^x and its corresponding cost and lateness are returned.

In the context of the drone finite-state machine, starting a trip, fulfilling requests, and replacing batteries are triggered by the respective action condition, resulting in a transition of the state. For instance, at the beginning of Algorithm 1, once the initial decision state S_0 is initialized, all drones $\delta \in \Delta$ are on the state "Standby". In this state, a drone δ waits for a trip to be assigned to it, which is done in line 4 of Algorithm 2. Then, once the drone finite-state machine is executed in line 6, the drone gets the first trip of the assigned schedule, and the state transitions to "Ready to fly".

During flight, the drone remains in this state until reaching a vertex, either a depot or a customer location. If the vertex is a customer, then the state changes to "Delivering", and stays on it for the duration of the service time η. After finishing the delivery, the state transitions

back to "Flying", and continues until it reaches the next vertex. If the vertex is the depot, then the drone has finished a trip, and its state transitions to "Idle", where it waits for its depleted battery to be swapped with a fully charged one. Once a new battery is assigned to the idle drone, its state transitions to "Swapping battery", which is done for the duration ρ. After the battery swapping, the drone state returns to "Standby" wherein it awaits the loading of all scheduled parcels before continuing with the next scheduled trip and repeats the transitions again. If no trip is left, then the drone waits for the next trip assignment of the next epoch. The empty state "End" is only reached at the end of the MDP execution.

5.2. Stochastic information and transition to pre-decision state

After the execution of an action x triggering the deterministic transition to the post-decision state S_k^x, the arrival of random exogenous information represented by set $\mathcal{U}_{k+1}$ determines the stochastic transition from S_k^x to S_{k+1}. The time associated with the new state is defined as $t_{k+1} = t_k + \Psi$, with Ψ denoting a given time step eventually calibrated based on the process describing the arrival of new requests during the day. Our choice differs from other contributions that associate the new decision point with the arrival of a new request. The motivation comes from the difference in the considered problems. During some hours of the day, the frequency of arrival of requests can be very high, making the continuous update of the routing plans useless.

The transition to the next decision state S_{k+1} starting from S_k^x entails updating the sets of requests, which is done in lines 6–11 of Algorithm 1. The set of new requests appearing between t_k and t_{k+1}, denoted by set $\mathcal{U}_{k+1}$, is updated in line 6 with the newly-arriving random requests. Line 7 updates the set of outstanding requests $\mathcal{D}_{k+1}$ with the requests from epoch k, i.e., $\mathcal{D}_{k+1} \leftarrow \mathcal{D}_k^x$. Finally, the set of drones Δ_{k+1} is updated to include the new states of drones for the next period encompassing their flying operations and/or recharge operations at the depot, and $\mathcal{B}_{k+1}$ is also updated to consider the new state of the batteries.

5.3. Algorithm components

This section provides a comprehensive, step-by-step description of the procedures invoked in Algorithm 2 to perform the action derived from the adopted CFA policy. As previously indicated, it is composed of four steps: a trip generation procedure, an algorithm to select trips maximizing the number of customers served, a procedure to assign trips to drones in order to

visit all customers previously selected while minimizing total costs (distance and delay), and finally a battery assignment to the drones that are scheduled to fly.

5.3.1. Trip Construction Heuristic

This section details the procedure *build_trips* called in line 2 of Algorithm 2. Let $\mathcal{G}_k = (\mathcal{N}_k, \mathcal{A}_k)$ be the undirected subgraph of G induced by the delivery locations known at the decision epoch t_k, besides the depot i_0, where $\mathcal{N}_k \subseteq \mathcal{N}$, $\mathcal{N}'_k = \mathcal{N}_k \setminus \{i_0\}$, and $\mathcal{A}_k \subseteq \mathcal{A}$. Formally, the node set is defined as $\mathcal{N}'_k = \{i_r \mid r = (i_r, q_r, l_r) \in \mathcal{D}_k \cup \mathcal{U}_k\}$ and the set of edges is defined as $\mathcal{A}_k = \{\{i_r, i_p\} \mid i_r, i_p \in \mathcal{N}_k\}$. To build trips, we propose a label-setting algorithm. This algorithm relies on a backward approach and considers energy saving. Let us consider a generic trip $\gamma = \{i_0, i_1, \ldots, i_r, \ldots, i_p, i_{p+1}\}$ visiting the sequence of customer nodes $\mathcal{P}_\gamma = \{i_1, \ldots, i_p\}$, where $i_0 = i_{p+1}$. We define the trip feasible if: *i*) the total payload is lower than the drone's capacity and *ii*) the state of charge of the battery at the end of the trip is greater than or equal to a given threshold E_{min}. Given that energy consumption is random, we employ the paradigm of chance constraints (CC) to build safe trips. In particular, we impose that:

$$\mathbb{P}(\widetilde{CE}_\gamma \geq E_{min}) \geq \alpha,$$

where $\widetilde{CE}_\gamma$ denotes the total energy consumption random variable, and α is a probability value in (0, 1) used to calibrate the risk attitude. In particular, the higher this value, the more risk-averse the decision maker is and the more conservative the corresponding solutions. We deal with the CC assuming that the random variables in calculating $\widetilde{CE}_\gamma$ follow a Normal distribution and are independent. Under this assumption, the CC admits a deterministic equivalent reformulation that can be easily derived (e.g., Prékopa (1970)):

$$\mathbb{E}[\widetilde{CE}_\gamma] \geq E_{min} + \Phi^{-1}(\alpha) SDV(\widetilde{CE}_\gamma),$$

where $\Phi^{-1}(\alpha)$ is the α-quantile of the Normal standard distribution function and SDV the standard deviation. In the formula, the expected total energy consumption and its standard deviation can be derived by exploiting the properties of the Normal random variables. Specifi-

cally,

$$\mathbb{E}\left[\widetilde{CE}_\gamma\right] = \mathbb{E}\left[\sum_{h=0}^{|\mathcal{P}_\gamma|-1} \tilde{e}_{i_h,i_{h+1}}(q_{i_h,i_{h+1}}) + \tilde{e}_{i_{|\mathcal{P}_\gamma|},i_0}\right] = \sum_{h=0}^{|\mathcal{P}_\gamma|-1} \bar{e}_{i_h,i_{h+1}}(q_{i_h,i_{h+1}}) + \bar{e}_{i_{|\mathcal{P}_\gamma|},i_0} v. \quad (4)$$

Here, $|\mathcal{P}_\gamma|$ denotes the number of served requests along the trip, whereas $\bar{e}_{i_h,i_{h+1}}(q_{i_h,i_{h+1}})$ is the expected energy consumption, that depends on the total payload $q_{i_h,i_{h+1}}$ on edge $\{i_h, i_{h+1}\}$ (including the drone weight). The last term in the sum accounts for the energy the drone consumes to fly back to the depot starting from the last location, considering only the drone weight v.

The load on a given edge $\{i_h, i_{h+1}\}$ also considers the payloads of the deliveries associated with nodes visited after i_h. In our case, we assume that the energy consumption is a linear function of the payload, and thus (4) can be rewritten as:

$$\mathbb{E}\left[\widetilde{CE}_\gamma\right] = \sum_{h=0}^{|\mathcal{P}_\gamma|-1} \bar{e}_{i_h,i_{h+1}} \left(\sum_{l=h+1}^{|\mathcal{P}_\gamma|} q_{i_l}\right) + \bar{e}_{i_{|\mathcal{P}_\gamma|},i_0} v.$$

The standard deviation, under the independence assumption, can be written as:

$$SDV\left[\widetilde{CE}_\gamma\right] = \sqrt{\sum_{h=0}^{|\mathcal{P}_\gamma|-1} \sigma^2_{i_h,i_{h+1}} \left(\sum_{l=h+1}^{|\mathcal{P}_\gamma|} q_{i_l}\right)^2 + \sigma^2_{i_{|\mathcal{P}_\gamma|},i_0} v^2},$$

with $\sigma^2_{i_h,i_{h+1}}$ and $\sigma^2_{i_{|\mathcal{P}_\gamma|},i_0}$ denoting the variance of the energy consumption over edges $\{i_h, i_{h+1}\}$ and $\{i_{|\mathcal{P}_\gamma|}, i_0\}$, respectively. To avoid complicating the notation, the dependence of the variance on the payload will not be explicitly displayed in the sequel.

The formulas introduced above are used in the trip construction phase. The trips are built by implementing a dynamic programming backward label setting algorithm with different strategies for node extensions by distance and urgency. In this heuristic, trips are constructed in a backward manner because we can compute a trip's exact payload and energy consumption while it is being constructed without the need for additional computational burden. In particular, for each node i_r explored during the trip construction we introduce two sets, $\mathcal{I}^d(i_r)$ and $\mathcal{I}^l(i_r)$. In the former, nodes i_p connected to i_r are sorted according to non-decreasing values of the

distance from i_r to i_p, whereas in the latter, according to non-decreasing $l_p - t_k$ values. In the first case, we are interested in minimizing distances, while in the second one, the aim is to prioritize the most urgent requests to minimize lateness.

The backward labeling heuristic, presented by Algorithm 3, aims to create trips fulfilling the chance constraint $\mathbb{P}(\widetilde{CE}_\gamma \geq E_{min}) \geq \alpha$ and generate trips that are optimized for the distance traveled or total lateness. Algorithm 3 finds a set of non-dominated trips, denoted as Γ_k. The domination rule states that if two distinct trips $\gamma(j_1) = \left\{i_0, i_1^1, \ldots, i_p^1, i_{p+1}\right\}$ and $\gamma(j_2) = \left\{i_0, i_1^2, \ldots, i_p^2, i_{p+1}\right\}$ are given, the latter is dominated by the former if the following conditions hold:

a) $\mathcal{P}_{\gamma(j_1)} = \mathcal{P}_{\gamma(j_2)}$;

b) $\sum_{h=0}^{|\mathcal{P}_{\gamma(j_1)}|-1} \bar{e}_{i_h^1 i_{h+1}^1} \left(\sum_{l=h+1}^{|\mathcal{P}_{\gamma(j_1)}|} q_{i_l^1}\right) \leq \sum_{h=0}^{|\mathcal{P}_{\gamma(j_2)}|-1} \bar{e}_{i_h^2 i_{h+1}^2} \left(\sum_{l=h+1}^{|\mathcal{P}_{\gamma(j_2)}|} q_{i_l^2}\right)$, i.e., the total average energy consumption of trip $\gamma(j_1)$ is not greater than that of trip $\gamma(j_2)$.

To build set Γ'_k, Algorithm 3 begins with a set Γ'_k, which is composed of trips that can be extended, i.e., trips where a customer may still be inserted. At the beginning of the algorithm, set Γ'_k starts with an empty trip $\gamma = \{i_0, i_{p+1}\}$. At each iteration of Algorithm 3, up to Σ nodes adjacent to the last backward-inserted node i_r are selected to be inserted in the current trip γ, creating up to Σ new trips. The set of vertices selected to be inserted in the current trip is denoted by $\mathcal{V}$ and is found in line 7 by Algorithm 4. If set $\mathcal{V}$ is empty, indicating that no node was found by the procedure in line 7, and thus that trip γ can no longer be backward extended, then γ is added to Γ_k, provided that γ is not dominated by any other trip in Γ_k. If set $\mathcal{V}$ is not empty, a new trip γ' is generated for each $i \in \mathcal{V}$ by inserting i backward into γ. Thereafter, all non-dominated trips created by this step are inserted into Γ'_k. These steps are reiterated until Γ'_k becomes empty, indicating that no trip can be further extended. If there are still vertices that are not visited by any trip in Γ_k, then the algorithm is executed again but only considering unvisited vertices.

The selection process for vertices to be inserted into a trip γ by the backward heuristic is illustrated in Algorithm 4. This method finds a set of $\mathcal{V}$ containing up to Σ vertices to be inserted in γ. Let i_r be the last backward-inserted vertex in the trip γ, and $i_p \in \mathcal{I}^d(i_r)$ (or $i_p \in \mathcal{I}^l(i_r)$). Vertex i_p is only added to $\mathcal{V}$ if its insertion in γ does not violate the drone's capacity and the maximum energy consumption, as defined by inequalities (5) and (6). We observe that

Algorithm 3: Backward heuristic (*build_trips*).

input : Set of outstanding and newly arrived requests $\mathcal{D}_k \cup \mathcal{U}_k$ at epoch k.
output : Set of trips Γ_k.

1 Define the maximum size of set $\mathcal{V}$, represented by parameter Σ
2 $\Gamma_k \leftarrow \emptyset$
3 $\Gamma'_k \leftarrow \{i_0, i_{p+1}\}$ `// set of trips being build. Initially, set` Γ'_k `has only trip` $\{i_0, i_{p+1}\}$
4 **while** $\Gamma'_k \neq \emptyset$ **do**
5 $\quad \gamma \leftarrow select\_trip(\Gamma'_k)$ `// select a trip from set` Γ'_k
6 $\quad \Gamma'_k \leftarrow \Gamma'_k \setminus \gamma$ `// remove` γ `from` Γ'_k
7 $\quad \mathcal{V} \leftarrow select\_vertices\_to\_insert(\gamma, \Sigma)$ `// select vertices by Algorithm 4`
8 $\quad$ **if** $\mathcal{V} = \emptyset$ **and** γ *is not dominated by any trip in* Γ_k **then**
9 $\quad\quad \Gamma_k \leftarrow \Gamma_k \cup \gamma$ `// add` γ `to the set of trips to be returned`
10 $\quad$ **else if** $\mathcal{V} \neq \emptyset$ **then**
11 $\quad\quad$ **foreach** $i \in \mathcal{V}$ **do** `// create` $|\mathcal{V}|$ `new trips by inserting each` i `in` γ
12 $\quad\quad\quad \gamma' \leftarrow \gamma \cup \{i\}$ `//` i `is inserted after the last backward-inserted vertex` i_r
13 $\quad\quad\quad \Gamma'_k \leftarrow \Gamma'_k \cup \gamma'$ `// add` γ `to the set of trips being build`
14 $\quad\quad$ **end**
15 $\quad$ **end**
16 **end**
17 **return** Γ_k

inequality (6) also accounts for the energy needed to return to the depot. If all vertices in the neighbor set of i_r (either $\mathcal{I}^d(i_r)$ or $\mathcal{I}^l(i_r)$) are explored or $|\mathcal{V}| = \Sigma$, then Algorithm 4 stops and returns the set $\mathcal{V}$ as it is, even if it is empty.

$$\sum_{l=1}^{|\mathcal{P}_\gamma|} q_{i_l} + q_{i_p} \leq Q \tag{5}$$

$$\bar{e}_{i_0 i_p}\left(q_p + \sum_{l=|\mathcal{P}_\gamma|}^{1} q_{i_l}\right) + \bar{e}_{i_p i_{|\mathcal{P}_\gamma|}}\left(\sum_{l=|\mathcal{P}_\gamma|}^{1} q_{i_l}\right) + \sum_{h=|\mathcal{P}_\gamma|}^{2} \bar{e}_{i_h i_{h-1}}\left(\sum_{l=h-1}^{1} q_{i_l}\right) + \bar{e}_{i_1 i_{n+1}} v \leq$$
$$E_{max} - \Phi^{-1}(\alpha)\sqrt{\sigma^2_{i_0 i_p} + \sigma^2_{i_p i_{|\mathcal{P}_\gamma|}} + \sum_{h=|\mathcal{P}_\gamma|}^{2} \sigma^2_{i_h i_{h-1}} + \sigma^2_{i_1 i_{n+1}}}. \tag{6}$$

5.3.2. Trip Selection Problem

This section provides a detailed description of the procedure *select_trips* invoked in line 3 of Algorithm 2. As the procedure presented in Section 5.3.1 finds several trips, selecting and assigning the most promising ones to the drones is necessary. Algorithm 5 depicts the procedure for selecting these trips. First, as we want to serve as many (urgent) requests as possible, we

Algorithm 4: *select_vertices_to_insert*.

input : Trip γ; Σ, the maximum size of set $\mathcal{V}$.
output : Set of vertices $\mathcal{V}$.

Let $i_r \in \gamma$ be the first location to be visited by the trip γ, after the depot i_0
$\mathcal{V} \leftarrow \emptyset$
while $|\mathcal{V}| < \Sigma$ **and** *there is a neighbor vertex of* i_r *to be explored* **do**
    `/* select` i_p `from the correct neighbor set, following the sorting policy */`
    $i_p \leftarrow select\_first\_neighbor(\mathcal{I}^d(i_r))$ **or** $i_p \leftarrow select\_first\_neighbor(\mathcal{I}^l(i_r))$
    if *inequalities* (5) *and* (6) *hold when inserting* i_p *into* γ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup i_p$
    end
    $\mathcal{I}^d(i_r) \leftarrow \mathcal{I}^d(i_r) \setminus i_p$ **or** $\mathcal{I}^l(i_r) \leftarrow \mathcal{I}^l(i_r) \setminus i_p$ `// remove` i_p `from the correct neighbor set`
end
return $\mathcal{V}$

solve a Set Packing-Based Problem (SPBP), which aims to maximize the number of (urgent) requests selected. Second, we use the requests served in the trips of an optimal solution of the SPBP as input for a non-overlapping Δ-Trip Set Selection Problem (DTSSP). The DTSSP aims to minimize non-overlapping trip routing costs while ensuring that urgent requests selected in the optimal solution of the SPBP are covered by at least one trip.

In Algorithm 5, the selected trips are not yet assigned to drones; rather, they are selected as candidates for an assignment because they are built by ignoring which drones are flying and which ones are idle. Instead, Algorithm 5 considers that all drones are available at the depot to make the models simpler and easier to solve. The method used to assign the chosen trips to the drones is detailed in Section 5.3.5. The mathematical formulations of the SPBP and DTSSP are shown in Sections 5.3.3 and 5.3.4, respectively.

Algorithm 5: Select trips (*select_trips*).

input : Set of outstanding and newly arrived requests $\mathcal{D}_k \cup \mathcal{U}_k$; Set of trips Γ_k; number of drones n; maximum number of trips per drone M.
output : Set of trips Γ_k^* without repeated requests.

$\mathcal{D}_k^{pr} \cup \mathcal{U}_k^{pr}, \mathcal{D}_k^{npr} \cup \mathcal{U}_k^{npr} \leftarrow split(\mathcal{D}_k \cup \mathcal{U}_k)$ `// split requests into priority` $\mathcal{D}_k^{pr} \cup \mathcal{U}_k^{pr}$ `and non-priority` $\mathcal{D}_k^{npr} \cup \mathcal{U}_k^{npr}$
$\mathcal{D}'_k \cup \mathcal{U}'_k \leftarrow$ SPBP$(\mathcal{D}_k^{pr} \cup \mathcal{U}_k^{pr}, \mathcal{D}_k^{npr} \cup \mathcal{U}_k^{npr}, \Gamma_k, n, M)$ `// solve model` (7a)−(7h) `and find set` $\mathcal{D}'_k \cup \mathcal{U}'_k$
$\Gamma_k^* \leftarrow$ DTSSP$(\mathcal{D}_k \cup \mathcal{U}_k, \mathcal{D}'_k \cup \mathcal{U}'_k, \Gamma_k, n, M)$ `// solve model` (8a)−(8b) `and find set of trips` Γ_k^*
return Γ_k^*

5.3.3. Set Packing-Based Problem

Let $\mathcal{D}_k^{pr} \cup \mathcal{U}_k^{pr} = \{r \mid r \in \mathcal{D}_k \cup \mathcal{U}_k \wedge l_r \in [0, \min\{t_k + \bar{l}, T\}]\}$ be the set of priority outstanding and newly arrived requests at epoch k, i.e., requests whose soft deadline lies within $\big[0, \min\{t_k + \bar{l}, T\}\big]$, where $\bar{l}$ is a non-negative value in minutes. It thus follows that the set $\mathcal{D}_k^{npr} \cup \mathcal{U}_k^{npr} = \{r \mid r \in \mathcal{D}_k \cup \mathcal{U}_k \wedge r \notin \mathcal{D}_k^{pr} \cup \mathcal{U}_k^{pr}\}$ is defined as the set of non-priority requests, that is, requests for which the delivery can be postponed to after $t_k + \bar{l}$. Moreover, consider $\Gamma_{k,r} \subseteq \Gamma_k$ as the set of trips containing request $r \in \mathcal{D}_k \cup \mathcal{U}_k$. The SPBP can be formulated with binary decision variables $y_{\gamma j}$ equal to one if trip $\gamma \in \Gamma_k$ is assigned to drone j, and z_r equal to one if customer $r \in \mathcal{D}_k \cup \mathcal{U}_k$ is visited by at least one trip. The problem can then be modeled as:

$$(\texttt{SPBP}) \ \max \sum_{r \in \mathcal{D}_k^{pr} \cup \mathcal{U}_k^{pr}} \mu^{pr} z_r + \sum_{r \in \mathcal{D}_k^{npr} \cup \mathcal{U}_k^{npr}} \mu^{npr} z_r \tag{7a}$$

subject to

$$\sum_{j=1}^{n} y_{\gamma j} \leq 1, \qquad \forall \gamma \in \Gamma_k \tag{7b}$$

$$\sum_{\gamma \in \Gamma_k} y_{\gamma j} \leq M, \qquad j = 1, \ldots, n \tag{7c}$$

$$z_r \leq \sum_{\gamma \in \Gamma_{k,r}} \sum_{j=1}^{n} y_{\gamma j}, \qquad \forall r \in \mathcal{D}_k \cup \mathcal{U}_k \tag{7d}$$

$$\sum_{\gamma \in \Gamma_k} \left(\sum_{h=0}^{|\mathcal{P}_\gamma|-1} \bar{t}_{i_h i_{h+1}} \right) y_{\gamma j} \leq \sum_{\gamma \in \Gamma_k} \left(\sum_{h=0}^{|\mathcal{P}_\gamma|-1} \bar{t}_{i_h i_{h+1}} \right) y_{\gamma j+1}, \qquad j = 1, \ldots, n-1 \tag{7e}$$

$$t_k + \sum_{\gamma \in \Gamma_k} \left(\sum_{h=0}^{|\mathcal{P}_\gamma|-1} \bar{t}_{i_h i_{h+1}} \right) y_{\gamma j} + \rho \left(\sum_{\gamma \in \Gamma_k} y_{\gamma j} - 1 \right) \leq T, \qquad j = 1, \ldots, n \tag{7f}$$

$$y_{\gamma j} \in \{0, 1\}, \qquad \forall \gamma \in \Gamma_k, \ j = 1, \ldots, n \tag{7g}$$

$$z_r \in \{0, 1\}, \qquad \forall r \in \mathcal{D}_k \cup \mathcal{U}_k. \tag{7h}$$

The objective function (7a) seeks to maximize the weighted number of customers visited on selected trips. In this context, the coefficients μ^{pr} and μ^{npr} represent the weights given to urgent and non-urgent customers. Each trip must be assigned to at most a single drone, following

constraints (7b). The number of trips assigned to a drone is limited to a maximum of M, as guaranteed by constraints (7c). Constraints (7d) link variables z_r to $y_{\gamma j}$ and ensure that z_r is equal to one only if customer r is contained in at least one selected trip. Constraints (7e) break solution symmetry concerning the drone indices $j = 1, \ldots, n$. Introducing these constraints ensures that the sets of trips are assigned to drones in an order that reflects the increasing duration of the trips. Constraints (7f) ensure that the total time required for all trips assigned to a drone does not exceed the time limit. We note that constraints (7f) do not consider the time for scheduling the sequence of trips assigned to the same drone. This aspect is addressed in Section 5.3.5. Finally, constraints (7g) and (7h) define the domain of the variables.

5.3.4. Non-overlapping Δ-Trip Set Selection Problem

This problem aims to construct as many sets of trips as the drones available at the depot while ensuring that no customer is visited more than once on a trip. To achieve this, we optimize the following mathematical formulation and verify that the feasibility condition is satisfied. Specifically, the `SPBP` maximizes the number of (urgent) requests to be served by selecting trips that may overlap. The selected requests are now fixed and used to solve another model to minimize the routing costs. Let $\Gamma'_k \subseteq \Gamma_k$ denote the set of trips in an optimal solution to the model (7a)–(7h). Furthermore, let $\mathcal{D}'_k \cup \mathcal{U}'_k \subseteq \mathcal{D}_k \cup \mathcal{U}_k$ be the set of requests contained in at least one trip of Γ'_k. Then, the `DTSSP` can be formulated as:

$$(\texttt{DTSSP}) \ \min \sum_{\gamma \in \Gamma_k} c(\gamma) \sum_{j=1}^{n} y_{\gamma j} \tag{8a}$$

$$\text{subject to (7b), (7c), (7e), (7f), (7g), and to}$$

$$\sum_{\gamma \in \Gamma_{k,r}} \sum_{j=1}^{n} y_{\gamma j} \geq 1, \qquad \forall r \in \mathcal{D}_k \cup \mathcal{U}'_k \tag{8b}$$

$$\sum_{\gamma \in \Gamma_k} y_{\gamma j} \leq \sum_{\gamma \in \Gamma_k} y_{\gamma j+1} + 1, \qquad j = 1, \ldots, n-1 \tag{8c}$$

$$\sum_{\gamma \in \Gamma_k} y_{\gamma n} \leq \sum_{\gamma \in \Gamma_k} y_{\gamma 1} + 1. \tag{8d}$$

Objective function (8a) minimizes the total routing costs of the selected trips. The covering constraints (8b) ensure that requests in $\mathcal{D}'_k \cup \mathcal{U}'_k$ must be contained in at least one selected trip.

Constraints (8c) aim to achieve a balance in utilizing drones. This is achieved by imposing that a drone can only be assigned p trips, with $p > 1$, if at least $p - 1$ trips were assigned to drone $j + 1$. It is essential to impose constraints (8d) to guarantee that the last drone, identified by index n, also complies with the balanced utilization of drones prescribed by constraints (8c). To accelerate the resolution of the mathematical formulation (8a)–(8d), the solution of the SPBP model is employed as a warm start. If the optimal solution of (8a)–(8c) contains trips that visit the same customer more than once, this visit is removed from the trip with the fewest customers visited. If the trip becomes empty as a result, it is eliminated.

5.3.5. Trip Assignment Problem

Let $\Gamma_k^* \subseteq \Gamma_k$ be the set of trips, without repeated requests, selected by solving the mathematical formulation (8a)–(8b), such that $\bigcup_{j=1}^{n} \Gamma_{k,j}^* = \Gamma_k^*$, where $\Gamma_{k,j}^*$ is the subset of trips that can be assigned to drone $j = 1, \ldots, n$. We observe that $\Gamma_{k,j_1}^* \cap \Gamma_{k,j_2}^* = \emptyset$ for $1 \leq j_1, j_2 \leq n$ and $j_1 \neq j_2$. Each subset $\Gamma_{k,j}^*$ does not consider the current status of drone j. Consequently, it could be a suboptimal decision to assign it to j. Instead, better assignments are identified using the procedure described in Algorithm 6, taking into account the least lateness of each trip schedule obtained with a permutation of the trips, and the state of the drones at the current time of the epoch. The trips assigned by Algorithm 6 will be executed by the drone upon its return to the depot, where it will also swap its battery.

Algorithm 6: Assign trips (*assign_trips*).

input : Current epoch time t_k; set of trips Γ_k^*; set of drones Δ_k.
output : Assignment of trip schedules to drones $\{\Gamma_k(\delta) : \delta \in \Delta_k\}$.

foreach $j = 1, \ldots, |\Delta|$ **do**
  foreach $\Gamma_{k,j}^* \in \Gamma_k^*$ **do** `// find the best permutation of each trip schedule`
    $\Gamma_{k,j}^* \leftarrow \bar{\Pi}(\Gamma_{k,j}^* t_k)$
    foreach $\delta \in \Delta_k$ **do** `// compute the the total lateness of each possible assignment`
      $\psi_{\Gamma_{k,j=\delta}^*} \leftarrow compute\_delay(\Gamma_{k,j=\delta}^*)$
    end
  end
end
$\{\Gamma_k(\delta) : \delta \in \Delta_k\} \leftarrow$ `TAP`$(\{\psi_{\Gamma_{k,j}^*} : \Gamma_{k,j}^* \in \Gamma_k^*, \delta \in \Delta\},\ \Delta_k)$ `// solve model` (9a)−(9d)
return $\{\Gamma_k(\delta) : \delta \in \Delta_k\}$

We denote by $\bar{\Pi}(\Gamma_{k,j}^*, t_k)$ the permutation operator that aims to find the best permutation

of the trips in each subset $\Gamma^*_{k,j}$, with $j = 1, \ldots, n$, to achieve the minimum lateness in the fulfillment of customer requests. The permutation operator $\bar{\Pi}(\Gamma^*_{k,j}, t_k)$ enumerates all possible trip schedules assigned to drone $j = 1, \ldots, n$ and selects the one in which the total lateness in the service of all requests of all trips, starting from the current time t_k, is minimum. Then, for each drone $\delta \in \Delta_k$, we compute the additional lateness (if any) that the trip schedule $\Gamma^*_{k,j=\delta}$ would have in case drone δ is still flying. In this case, the time required by δ to complete its trip and return to the depot must also be taken into account. Let $\psi_{\Gamma^*_{k,j=\delta}}$ denote the total lateness of the trip schedule $\Gamma^*_{k,j=\delta}$. This information, in combination with the trip schedules, is employed to solve the following Trip Assignment Problem (`TAP`):

$$(\texttt{TAP}) \ \min \sum_{j=1}^{|\Delta|} \sum_{\Gamma^*_{k,j} \in \Gamma^*_k} \sum_{\delta \in \Delta_k} \psi_{\Gamma^*_{k,j}} x_{\Gamma^*_{k,j}\delta} \tag{9a}$$

subject to

$$\sum_{\forall \delta \in \Delta_k} x_{\Gamma^*_{k,j}\delta} = 1, \qquad \Gamma^*_{k,j} \in \Gamma^*_k, \ j = 1, \ldots, n \tag{9b}$$

$$\sum_{j=1}^{n} \sum_{\Gamma^*_{k,j} \in \Gamma^*_k} x_{\Gamma^*_{k,j}\delta} = 1, \qquad \forall \delta \in \Delta_k \tag{9c}$$

$$x_{\Gamma^*_{k,j}\delta} \in \{0, 1\}, \qquad \Gamma^*_{k,j} \in \Gamma^*_k, \ j = 1, \ldots, n, \ \forall \delta \in \Delta_k \tag{9d}$$

where decision variables $x_{\Gamma^*_{k,j}\delta}$ are defined as follows:

$$x_{\Gamma^*_{k,j}\delta} = \begin{cases} 1, & \text{if the permutated trip schedule } \Gamma^*_{k,j} \text{ is assigned to drone } \delta \in \Delta_k, \\ 0, & \text{otherwise.} \end{cases}$$

Objective function (9a) seeks to minimize the overall lateness associated with the assignment of each permutation of trips to a drone in Δ_k. Constraints (9b) indicate that a drone is assigned to exactly a trip schedule, while constraints (9c) ensure that each trip schedule is assigned to a single drone. Variables $x_{\Gamma^*_{k,j}\delta}$ are binary, as defined in (9d), and define the action to be taken at epoch k. We observe that the solution provided by the `TAP` can assign a trip schedule to a drone where some trips can start at a time greater than or equal to $t_{k+1} = t_k + \Psi$. Using our CFA policy, these trips will be canceled in the next pre-decision state S_{k+1}, as their customer

requests can be reassigned and rescheduled more effectively in the next epochs, given that new requests appear. On the contrary, if the myopic policy $\pi^{LA}\,(M = +\infty)$ is used, all the trips assigned to a drone are executed and remain unchanged, even if they start in the next epochs.

5.3.6. Battery Assignment Procedure

After finishing a trip and returning to the depot, each drone is loaded with the deliveries to be performed on the next trip. It also undergoes a battery swap, replacing its depleted battery with a fully charged one. To ensure that the batteries are utilized uniformly and thus age in a homogeneous manner, it is necessary to monitor the number of times each battery β is employed. This is represented by ω_k^β. The batteries are sorted in a priority queue based on their usage count, with those used fewer times having higher priority. Therefore, upon the arrival of a drone at the depot, the first battery of this queue is removed and equipped to the drone. The battery swap is conducted within the drone finite-state machine, as illustrated in Figure 1.

Once a recharged battery is equipped on a drone, the former drone's battery is removed and not inserted into the priority queue immediately. Instead, the battery is first recharged. The time to fully recharge the battery depends on its remaining charge and the recharge ratio, expressed as $e'_\beta = 5\%/min$. Consequently, an empty battery requires 20 minutes to recharge. Upon completion of the recharging process, the battery becomes available to be equipped on a drone again and inserted into the priority queue. If this queue is empty when a drone arrives at the depot, the drone waits for a battery to finish its recharge, thus becoming available for use.

6. Computational results

In this section, we assess the performance of our solution methodology on instances adapted from the literature. All algorithms have been implemented in *C++* and compiled with the *g++* compiler, version 12.3.0. We used Gurobi's *C++* API v.11.0.2 for solving the integer programs. All tests were executed on a computer with an Intel® Core™ i9-13900K processor with 32 threads at 3.0 GHz and 128 GB of RAM.

We generated our instances from those of Ulmer and Thomas (2018), which we obtained after requesting them from the first author. They shared 400 instances with 500 customers, of which 200 were generated using a uniform distribution and 200 using a normal distribution on the customers' coordinates. From each set of 200 instances, we randomly selected 50 and adapted them

to be used in this work. The initial adaptation entailed the removal of customers deemed unreachable (as we consider only drones, while Ulmer and Thomas (2018) also considered vehicles). An unreachable customer is one whom a drone cannot serve, even when performing a round trip, because the energy required to reach it exceeds the battery's capacity. Subsequently, customers were sampled randomly, and three new instances were derived with $m \in \{200, 300, 400\}$, resulting in 300 instances. In these instances, each customer node r is associated with a random demand $q_r \in [0.3kg, 2.0kg]$, a service time $\eta = 3min$, a time a_r at which customer r appears in the system, and the customer's soft deadline l_r. The value a_r was randomly generated in the interval $(0, T - 240min]$, and we set $l_r = a_r + 240min$ following Ulmer and Thomas (2018). Furthermore, the instances with $m = 200$ were solved with $n = 12$ drones, those with $m = 300$ were solved using $n = 18$, and those with $m = 400$ were solved using $n = 24$. These instances and detailed results are available from `https://www.leandro-coelho.com/drone-routing/`.

Except for parameters M and Ψ, whose values may vary depending on the specific tests employed, all remaining parameters are based on the values outlined in Table 2, obtained from the literature or after a preliminary testing phase. Some values pertaining to drones and batteries were derived from the data presented in Dorling et al. (2017) and Ulmer and Thomas (2018). In Table 2, $\sigma_{\bar{\xi}}$ is the standard deviation of the drone's average speed as a percentage of the average speed and α is the confidence interval, both used in inequality (6) to compute the drone's maximum energy consumption; Σ is the maximum number of adjacent nodes to be inserted in trip used in Algorithm 3; and μ^{pr} and μ^{npr} are the objective coefficients used in model (7a)–(7h). The other parameters are described in Table 1.

Table 2: Parameters values.

										Drone data						Battery data				
Parameter	$\sigma_{\bar{\xi}}$	α	Σ	μ^{pr}	μ^{npr}	μ^{d}	μ^{l}	g	$\bar{l}$	$\bar{\xi}$	v	Q	m	p	z	b	ρ	e'_{β}	E_{min}	E_{max}
Value	2%	97%	5	0.8	0.2	1	5	$9.81N/kg$	$40min$	$24km/h$	$3kg$	$2.3kg$	6	$1.204kg/m^3$	$0.0064m^2$	$2n$	$20min$	$5\%/min$	10%	100%

Section 6.1 presents preliminary tests to assess the steps of our solution approach. Section 6.2 shows the results of the tests performed in the benchmark instances.

6.1. Preliminary tests

This section presents the results of preliminary tests conducted to evaluate the performance of the proposed algorithm. In these preliminary tests, 12 instances were used, with two instances

of each value of $m \in \{200, 300, 400\}$ selected for both normal and uniform distributions. To choose these 12 instances, we picked the first two instances of each size and each distribution. In particular, the outcomes of the tests conducted to identify the optimal parameter value of M are presented in Section 6.1.1, and Section 6.1.2 outlines the computational analysis performed to evaluate each of the main steps of our solution approach.

6.1.1. Tunning parameter M

Two values of Ψ were considered: $20min$ and $40min$. These values were fixed by varying $M \in \{1, 2, 3, 4\}$ and selecting the best values for the tests presented in Section 6.2.

Table 3 shows the average CFA policy results across the 12 instances for each value of $M = 1, 2, 3, 4$. In this table, m^s_{avg} is the average number of customers served, c_{avg} is the average cost of the solutions, ψ_{avg} is the average lateness, τ_{avg} is the average distance traveled, and t (s) is the average total execution time in seconds.

Table 3: Comparison of π^{CFA} using different values of parameter M.

	$\Psi = 20min$					$\Psi = 40min$				
M	m^s_{avg}	c_{avg}	ψ_{avg}	τ_{avg}	t (s)	m^s_{avg}	c_{avg}	ψ_{avg}	τ_{avg}	t (s)
1	299.00	1715.09	51.00	1460.09	11.34	271.42	2877.27	317.92	1287.69	7.67
2	298.08	1894.81	83.17	1476.52	40.45	280.42	3164.43	302.75	1287.34	25.63
3	297.67	2113.19	127.33	1478.98	93.22	271.08	3004.28	342.25	1293.03	65.22
4	296.92	2161.46	136.67	1478.13	143.57	269.92	3031.71	346.67	1298.37	99.42

As we can observe from the results presented in Table 3, $\pi^{CFA}(M = 1)$ yields the best values for all metrics when $\Psi = 20min$. Regarding the tests with $\Psi = 40min$, the configuration with $M = 2$ yields the best results in terms of both the average number of customers served and the total lateness. In consideration of these outcomes, we have selected $M = 1$ and $M = 2$, respectively for $\Psi = 20min$ and $\Psi = 40min$, to be employed in the tests conducted in Section 6.2. These values of Ψ are consistent with the flight duration of a drone engaged in the delivery of goods in a practical setting.

6.1.2. Evaluating action components

In this section, we evaluate the main steps of our algorithm. First, we analyze the Trip Construction Heuristic by assessing the two insertion policies presented in Section 5.3.1, namely "by distance" and "by urgency" represented by sets $\mathcal{I}^d(.)$ and $\mathcal{I}^l(.)$, respectively. Second, we evaluate the trip selection step by testing Algorithm 5 in comparison to an alternative version that excludes the `SPBP` procedure, instead solely addressing the `DTSSP`. To demonstrate the

importance of Algorithm 6, we evaluated a variant of our approach that omitted this phase. In this variant, the trip assignments to drones were conducted following the DTSSP solution. We employed $M = 1$ and $\Psi = 20min$ in all tests conducted in this section.

Table 4 compares the two insertion policies (by distance and by urgency) for building trips in the backward heuristic. The first six instances in this table have their customer locations uniformly distributed, and in the last six they are normally distributed. The results of both policies are presented, including the total number of generated trips ($|\Gamma|_{total}$), the average number of trips selected per epoch ($|\Gamma'|_{avg}$), the total lateness in minutes (ψ), and the total energy consumption in kilowatts-minute ($\widetilde{CE}$). Additionally, the total time spent in the heuristic generating trips through all epochs ($\mathrm{H}_{t(s)}$) is provided, as is the total execution time of solving the set-partitioning problem across all epochs ($\mathtt{SPBP}_{t(s)}$). Finally, the total running time of the whole execution (t (s)) is also included.

Table 4: Comparison of $\pi^{CFA}(M = 1)$ with different backward heuristic insertion policies.

		By distance							By urgency						
Instance	m	$\lvert\Gamma\rvert_{total}$	$\lvert\Gamma'\rvert_{avg}$	ψ	$\widetilde{CE}$	$\mathrm{H}_{t(s)}$	$\mathtt{SPBP}_{t(s)}$	t (s)	$\lvert\Gamma\rvert_{total}$	$\lvert\Gamma'\rvert_{avg}$	ψ	$\widetilde{CE}$	$\mathrm{H}_{t(s)}$	$\mathtt{SPBP}_{t(s)}$	t (s)
bccl1_ud_m200	200	1803	71.85	193	37.74	0.38	1.17	5.55	1442	103.00	163	32.66	0.16	0.70	2.54
bccl2_ud_m200		1409	72.30	182	38.27	0.42	0.75	5.51	1934	138.14	0	31.15	0.17	0.75	2.77
bccl1_ud_m300	300	3119	90.67	21	56.41	0.66	2.75	10.13	2224	158.86	7	50.47	0.31	2.29	7.10
bccl2_ud_m300		2793	129.44	248	56.51	0.67	5.22	21.07	2216	158.29	34	50.81	0.38	3.08	8.47
bccl1_ud_m400	400	2927	168.00	152	76.96	1.11	11.79	35.10	2962	211.57	172	67.22	0.64	7.91	22.13
bccl2_ud_m400		4220	131.41	311	76.22	1.02	14.49	28.58	2317	165.50	57	65.27	0.58	6.92	17.19
average		2711.83	110.61	184.50	57.02	0.71	6.03	17.66	2182.50	155.89	72.17	49.60	0.37	3.61	10.03
bccl1_nd_m200	200	2217	106.50	383	32.86	0.48	1.11	4.86	2745	196.07	0	29.98	0.19	0.79	2.65
bccl2_nd_m200		2371	84.27	521	33.41	0.32	0.98	7.12	1494	106.71	30	30.35	0.17	0.78	2.24
bccl1_nd_m300	300	3110	117.54	733	49.01	0.69	3.18	11.25	2692	192.29	0	43.80	0.32	2.74	7.99
bccl2_nd_m300		3517	134.58	397	49.82	0.76	3.71	15.24	3042	217.29	53	47.14	0.36	2.53	7.88
bccl1_nd_m400	400	4730	227.52	283	66.57	0.99	8.30	41.56	2833	202.36	69	63.26	0.66	13.58	31.82
bccl2_nd_m400		3416	190.27	343	63.61	1.20	12.53	30.40	4119	294.21	27	61.51	0.55	8.99	23.34
average		3226.83	143.45	443.33	49.21	0.74	4.97	18.40	2820.83	201.49	29.83	46.01	0.38	4.90	12.65

As we can observe from Table 4, using the urgency insertion policy reduces the total number of trips generated, compared to the "by distance" one. This is because, in the "by urgency" approach, the algorithm can serve customers faster, i.e., in fewer epochs, and thus, there are some epochs in which no trips need be generated. The average number of trips generated (when an epoch requires them) is thus larger, but this happens less often than in the policy driven by distance. Moreover, the urgency policy has resulted in more trips being generated on average at each epoch, as each trip is shorter and visits fewer customers. Consequently, more energy is used to reach these vertices, resulting in shorter trips.

As expected, the utilization of the urgency policy in the construction of trips reduces overall

lateness compared to the "by distance" one. Despite selecting more (shorter) trips per epoch when optimizing urgency, the average energy consumption per drone is less than that observed when optimizing distance. This is because, in the latter case, drones will be more heavily loaded and will, therefore, utilize their batteries to a greater extent. Consequently, the urgency policy demonstrates a reduction in energy consumption compared to the distance policy.

Regarding the running times, both versions and the whole algorithm are extremely fast, but the backward heuristic is observed to run quicker when utilizing the urgency policy. This phenomenon can be attributed to how customer data is stored. Upon becoming visible to the system, a customer is stored in a priority queue in which outstanding customers are sorted by an increasing deadline. Consequently, customers are already sorted according to urgency. On the other hand, utilizing the "by distance" insertion policy necessitates periodically sorting outstanding customers by distance. This explains the difference in the running time of the backward heuristic. Furthermore, the implementation of the urgency policy results in a reduction in the number of trips generated, which in turn leads to a decrease in the execution time for solving the SPBP across all epochs in comparison to the distance-based policy. The implementation of the urgency insertion policy results in a reduction in the time required for the trip-building and selection steps. Accordingly, this insertion policy was selected for implementation in all tests conducted in this work.

Table 5 illustrates the comparative tests between the two route selection strategies: first, solving only the DTSSP to select trips from the entire set generated, and second, solving SPBP and then the DTSSP, as presented in Algorithm 5. To test this algorithm without the SPBP step, we impose that all urgent customers must be visited at least once in the DTSSP instead of imposing that all customers visited in the SPBP solution must be visited in the DTSSP solution, as the SPBP solution is not generated in this case. The objective of these tests was to demonstrate the significance of Algorithm 5 in conjunction with both the SPBP and DTSSP. Besides the total lateness and the total execution time of our method, Table 5 presents the number of customers served (m^s), the total distance traveled by the drones divided by the number of customers served (τ/m^s), and the total execution time to solve the DTSSP through all epochs in seconds ($\texttt{DTSSP}_{t(s)}$).

Table 5 also demonstrates that configuring Algorithm 5 with the resolution of the SPBP and subsequently the DTSSP, in contrast to merely using the DTSSP is much more efficient.

Table 5: Comparison of $\pi^{CFA}(M = 1)$ using different trips selection strategies.

		DTSSP					SPBP + DTSSP				
Instance	m	m^s	ψ	τ/m^s	$\texttt{DTSSP}_{t(s)}$	t (s)	m^s	ψ	τ/m^s	$\texttt{DTSSP}_{t(s)}$	t (s)
bccl1_ud_m200	200	161	247	5.48	3.00	5.65	200	163	5.17	1.25	2.54
bccl2_ud_m200		152	245	5.77	4.17	6.13	199	0	5.34	1.40	2.77
bccl1_ud_m300	300	235	31	4.97	10.68	12.30	300	7	5.22	3.49	7.10
bccl2_ud_m300		240	252	5.19	18.77	17.72	300	34	5.21	3.69	8.47
bccl1_ud_m400	400	338	177	4.93	65.19	95.79	399	172	5.46	10.51	22.13
bccl2_ud_m400		341	317	4.62	48.15	96.91	390	57	5.44	6.99	17.19
average		244.50	211.37	5.16	24.99	33.11	298.00	72.17	5.31	4.56	10.03
bccl1_nd_m200	200	142	391	6.57	3.10	6.52	200	0	4.31	1.24	2.65
bccl2_nd_m200		142	522	5.79	2.60	7.29	200	30	4.39	0.86	2.24
bccl1_nd_m300	300	223	754	5.40	13.30	26.02	300	0	4.47	3.80	7.99
bccl2_nd_m300		230	420	5.13	13.21	20.36	300	53	4.49	3.91	7.88
bccl1_nd_m400	400	306	297	4.29	39.59	68.87	400	69	4.58	9.62	31.82
bccl2_nd_m400		300	365	4.55	23.27	59.84	400	27	4.35	10.43	23.34
average		223.83	450.59	5.29	15.85	31.48	300.00	29.83	4.43	4.98	12.65

The former approach results in solutions with significantly more customers visited and reduced total lateness, as better trips can be selected using the SPBP + DTSSP configuration. It is evident that, although the solutions resulting from the DTSSP visit fewer customers than those obtained from the SPBP + DTSSP, the latter configuration consistently yields better τ/m^s ratios in the majority of instances. Moreover, visiting more customers is the main objective of our algorithms. Furthermore, it can be observed that employing the SPBP to select trips before the DTSSP reduces the time required for the latter to be solved. This is achieved by ensuring that customers included in trips selected by the SPBP are visited in a solution of the DTSSP, thereby producing a tighter problem. For these reasons, the configuration of Algorithm 5 with first solving the SPBP and then the DTSSP, has been employed in all tests conducted for this work.

Table 6 presents a series of tests designed to assess the efficacy of the TAP in assigning trips to drones. Two versions of the proposed method were tested: without and using Algorithm 6. In the first one, the algorithm assigns trips to drones in the same order as in the solution found by the DTSSP. In the second one, we solve the TAP to optimally assign trips to drones. Table 6 presents a structure similar to that observed in the preceding tables.

As demonstrated in Table 6, the application of the TAP following the selection of trips is another key aspect of our proposed algorithm. The application of the TAP results in enhanced overall efficiency, characterized by reduced lateness and an increased number of customers served. Furthermore, due to the optimization of trip-drone assignments and the enhanced efficiency of customer service, the overall running time of the algorithm is reduced. In light of these observations, the proposed algorithmic approach, as outlined in Algorithm 6, is employed in all tests

Table 6: Comparison of $\pi^{CFA}(M = 1)$ using or not the Algorithm 6 to assign trips to drones.

Instance	m	Without Algorithm 6				With Algorithm 6			
		m^s	ψ	$\|\Gamma\|_{total}$	t (s)	m^s	ψ	$\|\Gamma\|_{total}$	t (s)
bccl1_ud_m200	200	171	598	1868	4.07	200	163	1442	2.54
bccl2_ud_m200		171	160	1952	3.50	199	0	1934	2.77
bccl1_ud_m300	300	277	259	2448	12.23	300	7	2224	7.10
bccl2_ud_m300		276	624	3495	17.16	300	34	2216	8.47
bccl1_ud_m400	400	348	278	4536	59.68	399	172	2962	22.13
bccl2_ud_m400		344	252	3548	56.51	390	57	2317	17.19
average		264.50	361.83	2974.50	25.53	298.00	72.17	2182.50	10.03
bccl1_nd_m200	200	181	327	2769	4.30	200	0	2745	2.65
bccl2_nd_m200		180	209	2191	3.40	200	30	1494	2.24
bccl1_nd_m300	300	272	49	3056	16.31	300	0	2692	7.99
bccl2_nd_m300		283	143	3499	16.68	300	53	3042	7.88
bccl1_nd_m400	400	374	100	6143	66.85	400	69	2833	31.82
bccl2_nd_m400		384	404	4947	54.05	400	27	4119	23.34
average		279.00	205.33	3767.50	26.93	300.00	29.83	2820.83	12.65

presented in this work.

6.2. Numerical results

This section presents the results of the tests performed on all 300 benchmark instances. We compare our method using two settings: $\pi^{CFA}(M = +\infty)$, that is, a myopic approach where trips are not canceled, and $\pi^{CFA}(M = M^*)$, where we have used $M^* = 1$ for $\Psi = 20min$ and $M^* = 2$ for $\Psi = 40min$, calibrated after a simulation procedure described earlier.

Table 7 illustrates the outcomes of experiments conducted with $\Psi = 20min$. In this table, m^s_{avg} denotes the average number of customers served, m^{swl}_{avg} the average number of customers served without delay, and c_{avg} is the average solution cost, which is composed by the average total lateness (ψ_{avg}) and by the average total distance traveled (τ_{avg}). Furthermore, column t (s) shows the algorithm's execution time.

Table 7: Average results of tests with $\Psi = 20min$.

m	$\pi^{CFA}(M = +\infty)$						$\pi^{CFA}(M = 1)$					
	m^s_{avg}	m^{swl}_{avg}	c_{avg}	ψ_{avg}	τ_{avg}	t (s)	m^s_{avg}	m^{swl}_{avg}	c_{avg}	ψ_{avg}	τ_{avg}	t (s)
200	192.25	190.52	1308.80	75.21	932.75	1.92	199.76	198.70	1152.25	32.28	990.85	3.13
300	287.78	285.21	1958.13	107.81	1419.08	9.68	299.63	298.06	1681.66	41.13	1476.01	9.99
400	382.96	379.19	2971.99	212.60	1908.99	26.91	399.19	396.36	2319.13	75.42	1942.03	24.16

We can observe that the CFA policy with $M = 1$ dominates the myopic policy that does not limit the number of trips assigned to drones, which corresponds to $M = +\infty$. For all sizes of instances, the number of customers served is larger in our calibrated CFA policy. In particular, in the dataset with 200 customers, the average total cost of the CFA policy shows

a gap improvement of 11.96% in comparison to that of the myopic policy, computed as $100 \cdot (c_{avg}(M = +\infty) - c_{avg}(M = 1))/c_{avg}(M = +\infty)$. Additionally, the average lateness has a 57% lower gap in the CFA policy, calculated as $100 \cdot (\psi_{avg}(M = +\infty) - \psi_{avg}(M = 1))/\psi_{avg}(M = +\infty)$. In the data set with 300 customer demands, the average total cost exhibits a 14.12% reduction in the gap when compared to the myopic policy, while the average lateness demonstrates a notable decrease of 61.85% in comparison to the myopic policy. Finally, in the data set with 400 customer demands, the average total cost is observed to exhibit a reduction of 21.97% in the gap, while the average lateness shows a decrease of 64.52% in the gap in comparison to that observed in the myopic policy.

Table 8 presents the test results using $\Psi = 40min$. This table follows the same structure as Table 7, and shows a similar pattern of results. Specifically, the data set with 200 customer demands shows a reduction in the gap of 2.82% and 8.35% for the average total cost and average lateness, computed as $100 \cdot (c_{avg}(M = +\infty) - c_{avg}(M = 2))/c_{avg}(M = +\infty)$ and $100 \cdot (\psi_{avg}(M = +\infty) - \psi_{avg}(M = 2))/\psi_{avg}(M = +\infty)$, respectively. For the data set with 300 customer demands, we observe a reduction in the average total cost and average lateness, with a decrease of 1.15% and 2.93%, respectively. In the data set with 400 customer demands, we observe a minimal increase of 0.8% in the gap between the average total cost for the CFA policy and the myopic policy. In addition, we observe a decrease of 2.22% in the average lateness gap.

Table 8: Average results of tests with $\Psi = 40min$.

	$\pi^{CFA}(M = +\infty)$						$\pi^{CFA}(M = 2)$					
m	m^s	m^{swl}_{avg}	c_{avg}	ψ_{avg}	τ_{avg}	t (s)	m^s	m^{swl}_{avg}	c_{avg}	ψ_{avg}	τ_{avg}	t (s)
200	171.27	167.55	1777.00	183.98	857.10	8.59	188.34	184.33	1726.85	168.61	862.60	3.87
300	258.38	252.73	2656.29	273.91	1286.74	42.20	282.24	276.04	2625.75	265.89	1296.30	18.44
400	344.08	335.38	3700.42	394.62	1727.32	130.59	375.78	367.33	3730.05	385.86	1734.00	56.48

The analysis of the average routing cost in columns τ_{avg} of Tables 7 and 8 shows that the CFA policy might seem to perform slightly worse than the myopic policy. There are two explanations for this observation. First, recall that the first objective of our algorithm is to serve more customers. Our CFA policy excels in this criterion. Second, the lack of service to all customers, as evidenced in columns m^s_{avg}, can be attributed to the reduced flight range of the drones, which is influenced by the uncertainty in power consumption of the batteries and the limited number of drones available to complete all deliveries.

The box plots depicted in Figures 2 and 3 provide a comprehensive statistical overview of the performance of the CFA policy in comparison to the myopic one when $\Psi = 20min$. Figure 2 displays box plots of the pairwise percent increase in the average number of served customers achieved by

$$100 \cdot \frac{m^s_{M^*} - m^s_{M=+\infty}}{m^s_{M=+\infty}}, \tag{10}$$

where $m^s_{M^*}$ and $m^s_{M=+\infty}$ are the number of served customers by $\pi^{CFA}(M = M^*)$ and $\pi^{CFA}(M = +\infty)$, respectively. Recall that in tests with $\Psi = 20min$, we have used $M^* = 1$.

Figure 3 displays box plots of the pairwise percent reduction in average cost per served customer achieved by $\pi^{CFA}(M = 1)$ versus $\pi^{CFA}(M = +\infty)$ policy as computed by

$$100 \cdot \frac{\frac{c_{M=+\infty}}{m^s_{M=+\infty}} - \frac{c_{M^*}}{m^s_{M^*}}}{\frac{c_{M=+\infty}}{m^s_{M=+\infty}}}, \tag{11}$$

where c_{M^*} and $c_{M=+\infty}$ refer to the solution cost obtained by $\pi^{CFA}(M = M^*)$ and $\pi^{CFA}(M = +\infty)$, respectively.

Figure 2: Percent increase in average number of served customers by $\pi^{CFA}(M = 1)$ versus $\pi^{CFA}(M = +\infty)$.

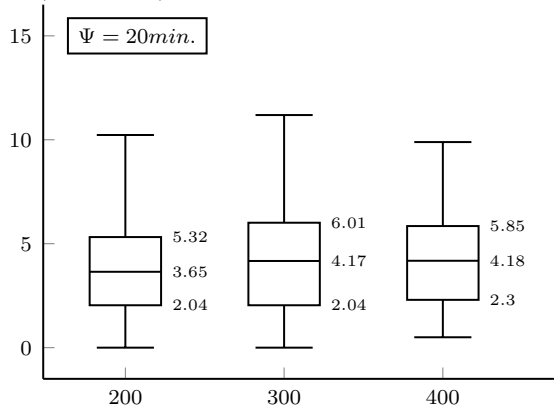
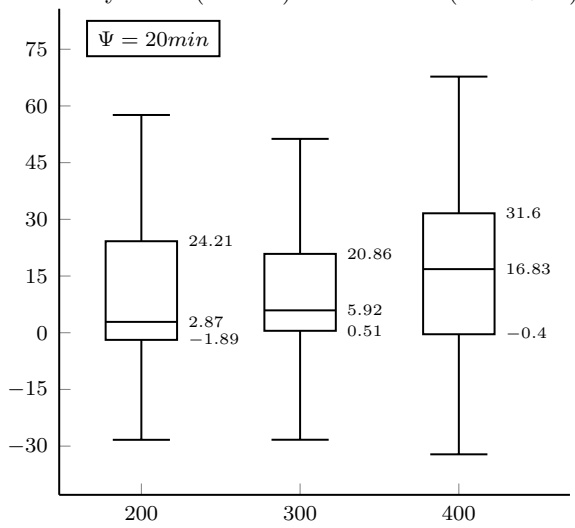


Figure 3: Percent reduction in average cost per served customer by $\pi^{CFA}(M = 1)$ versus $\pi^{CFA}(M = +\infty)$.



One can note from the box plot of Figure 2 that for the instances with 200 customer demands, the CFA policy achieved a median percent increase of 3.65%, 4.17% in the instances with 300 customer demands, and 4.18% in the instances with 400 customer demands. The 25th percentile of the percent increase values is positive for all the data sets, demonstrating that the CFA policy consistently achieves a greater total number of served customers in all instances.

Regarding the box plot of Figure 3, for all the instances, we observe a large variability in the average cost per served customer in the solutions provided by the CFA policy compared to those obtained with the myopic policy. The CFA policy achieves a median percent reduction of 2.87% in the data set with 200 customer demands, 5.92% in the data set with 300 customer demands, and 16.83% in instances with 400 customer demands.

Figures 4 and 5 show a statistical overview of the performance of the CFA policy in comparison to the myopic one using $\Psi = 40min$. Figure 4 depicts box plots of the pairwise percent increase in average number of served customers achieved by $\pi^{CFA}(M = 2)$ versus $\pi^{CFA}(M = +\infty)$ as computed by expression (10). Similarly, Figure 5 shows the equivalent information for the reduction in average cost per served customer, computed using expression (11). As it can be noted from Figure 4, for the instances with 200 customers, the CFA policy achieves a median percentage increase of 9.94%, of 9.54% for instances with 300 customers, and 9.31% for those with 400 customers. Unlike the box plots for the case $\Psi = 20$ minutes, we observe an almost constant trend of the median percentage increase of served customers. The 25th percentile of the percent increase values are positive for all the data sets, indicating that the CFA policy consistently results in a greater total number of served customers.

Figure 4: Percent increase in average number of served customers by $\pi^{CFA}(M = 2)$ versus $\pi^{CFA}(M = +\infty)$.

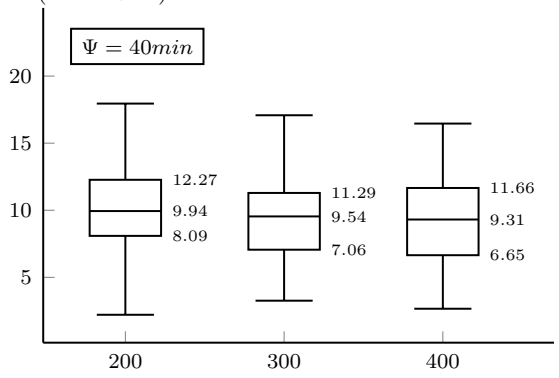
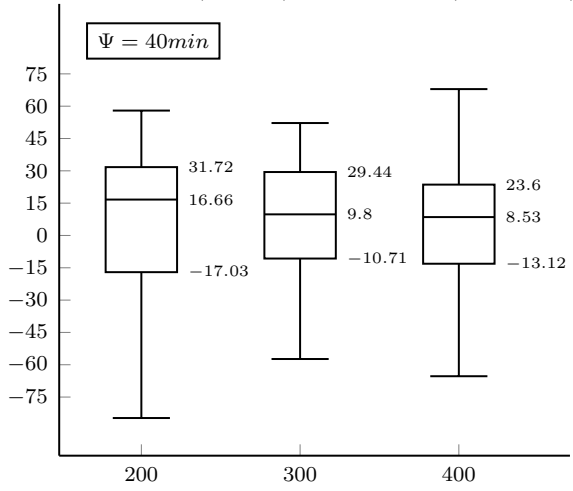


Figure 5: Percent reduction in average cost per served customer by $\pi^{CFA}(M = 2)$ versus $\pi^{CFA}(M = +\infty)$.



Considering results presented in the box plot of Figure 5, for all the instances, we observe a large variability in the average cost per served customer in the solutions provided by the CFA policy compared to those obtained with the myopic policy. The CFA policy achieves a median percent reduction of 16.66% in the data set with 200 customer demands, 9.8% in the data set

with 300 customer demands, and 8.53% in instances with 400 customer demands. In contrast to the case where $\Psi = 20$ minutes, we observe a decreasing trend for the median percentage reduction in cost per served customer.

7. Conclusions

This study considers the operational optimization problem of a fleet of drones deployed daily to make customer deliveries. The inherent uncertainty of this operation arises from the variability of customer demands and the drones' battery energy consumption. The objective is to design a system to plan trips and optimize their assignment to drones in a dynamic setting, considering the evolving delivery requirements throughout the day.

To form our real-time policies to assign and schedule trips fulfilling the demands, we have designed a tailored CFA policy that accounts for a restricted number of trips to be assigned to the drones. This ensures that the drones are ready at the depot to fulfill new requests that may arise during the day. Our computational results demonstrate that the proposed CFA policy achieves lower-cost solutions than a myopic policy. Further, we show that the CFA policy significantly reduces the lateness in serving the customers within their time windows compared to the myopic policy.

Our model and solution approach can accommodate the same or similar problems in which the uncertainty affects the travel time. This includes, for instance, multi-modal transport systems that employ autonomous robots for delivery purposes. Further research is required to compare this with different delivery forms in a dynamic context.

References

X. Chen, M. W. Ulmer, and B. W. Thomas. Deep q-learning for same-day delivery with vehicles and drones. *European Journal of Operational Research*, 298(3):939–952, 2022.

X. Chen, T. Wang, B. W. Thomas, and M. W. Ulmer. Same-day delivery with fair customer service. *European Journal of Operational Research*, 308(2):738–751, 2023.

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

C. Cheng, Y. Adulyasak, L.-M. Rousseau, and M. Sim. Robust drone delivery with weather information. *Optimization-online.org*, 139:364–387, 2020b. URL `http://www.optimization-online.org/DB_HTML/2020/07/7897.html`.

B. N. Coelho, V. N. Coelho, I. M. Coelho, L. S. Ochi, R. Haghnazar K., D. Zuidema, M. S. Lima, and A. R. da Costa. A multi-objective green uav routing problem. *Computers & Operations Research*, 88:306–315, 2017.

K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017.

S. French. How long do lipo batteries last, and how do i know if an old lipo battery is safe to use?, 2021. URL `https://www.thedronegirl.com/2020/05/24/lipo-batteries-last/`.

Y. Gao, S. Zhang, Z. Zhang, and Q. Zhao. The stochastic share-a-ride problem with electric vehicles and customer priorities. *Computers & Operations Research*, 164:106550, 2024.

V. Garg, S. Niranjan, V. Prybutok, T. Pohlen, and D. Gligor. Drones in last-mile delivery: A systematic review on efficiency, accessibility, and sustainability. *Transportation Research Part D: Transport and Environment*, 123:103831, 2023.

G. Macrina, L. Di Puglia Pugliese, F. Guerriero, and G. Laporte. Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120:102762, 2020.

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

A. Prékopa. On probabilistic constrained programming. In *Proceedings of the Princeton Symposium on Mathematical Programming*, pages 113–138, Princeton, NJ, 1970. Princeton University Press.

L. D. P. Pugliese, F. Guerriero, and M. G. Scutellá. The last-mile delivery process with trucks and drones under uncertain energy consumption. *Journal of Optimization Theory and Applications*, 191:31–67, 2021.

A. Ruszczyński and A. Shapiro. *Stochastic Programming, Handbook in Operations Research and Management Science*. Elsevier Science, Amsterdam, 2003.

N. Soeffker, M. U. Ulmer, and D. C. Mattfeld. Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research*, 298(3):801–820, 2022.

M. W. Ulmer and M. Savelsbergh. Workforce scheduling in the era of crowdsourced delivery. *Transportation Science*, 54(4):1113 – 1133, 2020.

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

M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and M. Hennig. Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests. *Transportation Science*, 53(1):185 – 202, 2019.

M. W. Ulmer, B. W. Thomas, A. M. Campbell, and N. Woyak. The restaurant meal delivery problem: Dynamic pickup and delivery with deadlines and random ready times. *Transportation Science*, 55(1):75 – 100, 2021.

R. van Steenbergen, M. Mes, and W. van Heeswijk. Reinforcement learning for humanitarian relief distribution with trucks and uavs under travel time uncertainty. *Transportation Research Part C: Emerging Technologies*, 157:104401, 2023.

J. Zhang and T. V. Woensel. Dynamic vehicle routing with random requests: A literature review. *International Journal of Production Economics*, 256:108751, 2023.

J. Zhang, J. F. Campbell, D. C. Sweeney II, and A. C. Hupman. Energy consumption models for delivery drones: A comparison and assessment. *Transportation Research Part D: Transport and Environment*, 90:102668, 2021.