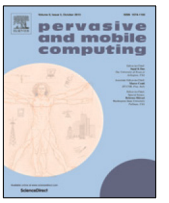
Pervasive and Mobile Computing 87 (2022) 101700

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc



# On the Scheduling of Conflictual Deliveries in a last-mile delivery scenario with truck-carried drones ☆

Francesco Betti Sorbelli [a], Federico Corò [a], Sajal K. Das [b], Lorenzo Palazzetti [c,*], Cristina M. Pinotti [a]

[a] Department of Computer Science and Mathematics, University of Perugia, Italy
[b] Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA
[c] Department of Computer Science and Mathematics, University of Florence, Italy

ARTICLE INFO

*Article history:*
Received 25 May 2022
Received in revised form 16 September 2022
Accepted 30 September 2022
Available online 5 October 2022

*Keywords:*
Drone-delivery
Multiple drones
Truck
Last-mile delivery system
Optimization

ABSTRACT

In this paper, we investigate the symbiosis between a truck and multiple drones in a last-mile package delivery scenario, introducing the Scheduling Conflictual Deliveries Problem (SCDP). From the main depot, a truck takes care of transporting a fleet of drones that will be used to deliver packages to customers. The route of the truck is predefined. Each delivery is associated with the energy cost of a drone, a reward that characterizes the priority of the delivery, and an interval between two points of the truck's route: the point from which the drone departs (launch point) and the point at which the drone returns to the truck (rendezvous point). The objective of the SCDP is to find a scheduling for the drones that maximizes the overall reward subject to the drone's battery capacity while ensuring that the same drone performs deliveries whose delivery intervals do not intersect. After showing that SCDP is an NP-hard problem, we devise an Integer Linear Programming (ILP) formulation for it. Furthermore, we devise a pseudo-polynomial time optimal algorithm for the single drone case and additional approximation algorithms for both the single and multiple drones case. Finally, we thoroughly compare the performances of our presented algorithms on different synthetic datasets.



## 1. Introduction

Since the Unmanned Aerial Vehicles (*drones*) have been developed, both the research and industrial communities have started to exploit this new technology in a plethora of applications. In particular, drones can be efficiently and effectively used for search and rescue operations over disastrous areas hit by an earthquake [1,2], for observing and monitoring crops in precision agriculture contexts [3–5], for localizing and monitoring missing people [6], or for shipping parcels to customers to accelerate the delivery process in a package delivery context [7–9]. Interestingly, the collaboration between drones and trucks, relying on their respective capabilities and mobility, can significantly improve the quality of service, especially in the last-mile delivery scenario [10,11].

In this paper, we investigate the cooperation between a *truck* and one or multiple drones for last-mile package delivery. With the help of drones, delivery companies can perform more deliveries and hence extend their revenue and

☆ This work was supported in part by the "GNCS – INdAM", by "HALY-ID" project funded by the European Union's Horizon 2020 under grant agreement ICT-AGRI-FOOD no. 862665, no. 862671, and by MIPAAF, Italy.

* Corresponding author.
*E-mail address:* lorenzo.palazzetti@unifi.it (L. Palazzetti).

https://doi.org/10.1016/j.pmcj.2022.101700
1574-1192/

business [12]. In fact, drones can deliver small packages quickly as they can easily traverse difficult terrain [13], often using shorter routes not otherwise possible for trucks. In this scenario, the drones fly (starting from the truck) to the assigned locations and deliver the packages, then leave and meet again with the truck to perform new deliveries. However, significant challenges arise due to such constraints as the drones' high energy consumption and their *current* inability to simultaneously serve multiple customers at a time [14,15]. The premise under consideration is that a delivery company has to make several deliveries to the customers in a city by relying on a truck carrying a fleet of drones with the same capabilities.

In this paper, we study this problem by assuming that the main depot knows the locations of the customers to visit and the road to travel for the truck. Therefore, before leaving the depot, the delivery company plans the drones' flying sub-routes to accelerate the deliveries, considering not only the energy used by the drones but also the revenue generated. A sub-route is uniquely identified by a starting and a rendezvous point on the truck's route. Since drones have limited battery capacity, *a delivery has a cost* in terms of the energy required, and this limits the number of deliveries that a drone can perform. Also, *conflicts among deliveries* can occur if they cannot be accomplished by the same drone, i.e., if the corresponding sub-routes, and hence their delivery intervals, intersect each other (see Fig. 1(a)). In addition, *each delivery has an associated reward* that characterizes its importance (e.g., a larger reward means higher priority). So, given the truck's route in the city, *the goal is to plan a scheduling for the drones such that the total reward is maximized subject to the constraints of limited drone's battery and the conflicts among deliveries*.

### 1.1. Motivations

The existing works in this area mainly study the problem of scheduling deliveries without considering neither any budget constraint for the drones nor possible conflicts among deliveries [16–18]. The absence of such details makes it harder to model real-world scenarios. Moreover, the proposed solutions in drone-based delivery applications are often determined by heuristic or meta-heuristic algorithms which do not provide any guaranteed results, in terms of solution's goodness [19,20]. In contrast, in this paper, we incorporate the aforementioned constraints in a last-mile delivery scenario, while also providing optimal, approximation, and heuristic solutions for our problem.

Another interesting aspect concerns the fact that the truck's route is already given. It is reasonable to assume that we know the route of the truck in advance for various reasons. For example, there can be a truck's route that is efficient most of the time considering traffic, road condition, and safety, and by using such a route, truck drivers can conduct their everyday delivery tasks comfortably and safely. Given the truck's route, we envision increasing the number of performed deliveries by exploiting a fleet of drones and reaching additional customers that were not first considered during the route planning. Our objective is not to change the route of the truck to serve more customers, but rather to serve the largest number of customers in the neighborhood of the truck's route. Therefore, the emphasis of our work is to possibly increase the number of deliveries without modifying the truck's route.

This work significantly extends our prior conference paper [21], by presenting new optimal and approximation algorithms along with a thorough analysis concerning their correctness. In particular, our results are summarized as follows:

- We define a novel optimization problem, called Scheduling Conflictual Deliveries Problem (SCDP), and prove it to be *NP*-hard. We also propose an Integer Linear Programming (ILP) formulation for SCDP.
- For the single drone case, we design optimal and approximation algorithms, while for the multiple drones case, we propose approximation and heuristic algorithms.
- We evaluate the performances of our algorithms on synthetic datasets.

The remainder of this paper is structured as follows. Section 2 surveys the related works. Section 3 introduces the SCDP showing its *NP*-hardness, and proposes an ILP formulation. Section 4 presents algorithms for solving SCDP with a single drone, while Section 5 devises algorithms for solving SCDP with multiple drones. Section 6 evaluates the algorithms on synthetic datasets. Finally, Section 7 offers conclusions and future research directions.

## 2. Related work

This section reviews the literature related to the problem of delivering packages with a truck carrying one or multiple drones, categorizing the works based on whether the deliveries are scheduled to a single or multiple drones. We recall that our proposed model considers both scenarios.

### 2.1. Single drone deliveries

The truck–drone cooperation in the last-mile delivery has been investigated for the first time when the flying sidekicks traveling salesman problem (FSTSP) has been introduced in [16]. The FSTSP is a particular case of the TSP, where drones take off from the truck, fly to deliver packages to customers, and go back to the truck in another place. Both vehicles can perform deliveries, but each has to stop waiting for the other at the rendezvous location. The authors propose a mixed-integer linear programming formulation (MILP) and two heuristic solutions for solving the FSTSP. Differently from

us, they do not take into account any reward and conflicts among deliveries, also because the truck and the drone can do multiple trips to/from the depot.

In the work [17], a single drone carried by a truck performs all the required deliveries. Differently from us, once any delivery has been performed, a drone can immediately recharge its battery. This means that all deliveries are feasible and, accordingly, no scheduling is required.

The heuristic algorithm proposed in [19] initially focuses on the truck's route and then improves the solution by considering the possible drone's sub-flights. A TSP solution for the truck only that fixes the order in which all the customers are served, is at first computed. Then, short drone sub-routes are greedily created by excluding some customers from the truck's route in order to reduce the overall makespan (i.e., the time difference between the start and end of a delivery sequence). However, the truck or the drone has to wait for the other at the rendezvous positions. Also, as we do, the truck is not allowed to go back on its route.

In the delivery system in [22], a truck carries a drone to deliver packages to customers. The objective is to minimize the total tour duration to serve all the customers. Differently from us, here there can be incompatible customers for the drone (e.g., package too heavy, or customer too far). So, the authors propose a MILP formulation based on timely synchronization of the truck and the drone trajectories. They also introduce a dynamic programming recursion based on an exact branch-and-price approach capable of optimally solving small instances.

### 2.2. Multiple drones deliveries

The investigated problem of delivering goods is further extended by considering also a fleet of drones, and numerous research works have been published.

In [20], the authors consider a scenario with multiple rechargeable drones. Drones can only carry one package at a time and have to return to the truck's roof to charge their battery after each delivery. The speed is the same for both drones and trucks, but their mobility metric is different (Euclidean metric for drones, Manhattan metric for the truck). The authors present a heuristic to solve the problem of finding a good schedule for the truck and all the drones that minimizes the average delivery time of the packages. In our paper, we do not consider the replenishment of batteries for drones, and also we optimize the reward and not the average delivery time.

A similar work to ours is proposed in [23]. The authors consider a predefined truck's route, and the problem is to optimize the planning of drones to/from the truck while serving the customers. However, unlike us, they need to determine the launch and meeting points between drones and the truck, while in our work these points are given in input. Furthermore, these points are calculated in such a way that the intervals generated do not intersect with each other, thus making all deliveries conflict-free. Importantly, the authors assume that drones do not have battery constraints. While their goal is to reduce total makespan, our goal is to maximize the reward for deliveries.

The delivery with a truck and drones with a rechargeable station is presented in [18]. The station can recharge multiple drones, and it is located near customer areas and away from the main depot. After having determined the lower bound of the number of drones that the station can handle, the authors show that this problem is a combination of TSP and parallel identical machine scheduling problems. Through this approach, they successfully reduce the complexity of the problem and obtain an exact solution. Differently from us, their goal is to minimize the overall makespan, under the assumption that the deliveries are equally relevant (each delivery has the same reward), and that all the deliveries are conflict-free.

A hybrid truck–drones scenario is presented in [24], where the authors, differently from us, propose to simultaneously employ trucks, truck-carried drones, and independent drones to construct a more efficient truck–drone parcel delivery system. A novel routing and scheduling algorithm is proposed to solve the hybrid parcel delivery problem.

In [25] a clustering approach is proposed to find locations for a truck in which it stops and deploys drones to deliver packages to near locations. In these cases, the total delivery time is determined by the longest flight time among drones in each cluster. The authors aim at minimizing the total delivery time by having a truck moving among the centers of clusters as drones operate in each cluster. However, all the deliveries can be performed since conflicts among them are not contemplated.

Last but not least, the seminal FSTSP has been extended, by the same authors in [26], for multiple drones (mFSTSP). A MILP formulation only suitable for small inputs, as well as faster heuristics for any input, have been devised to reduce the overall makespan.

## 3. Problem formulation

In this section, we first introduce the problem's underlying assumptions, then present the system and delivery models, and formally define the Scheduling Conflictual Deliveries Problem (SCDP). We also show that SCDP is *NP*-hard.

### 3.1. Problem assumptions

In this paper, we make a few problem assumptions listed as follows:

- The drones deliver a single package at a time due to stringent payload constraints. In other words, a drone cannot carry more than a single package.

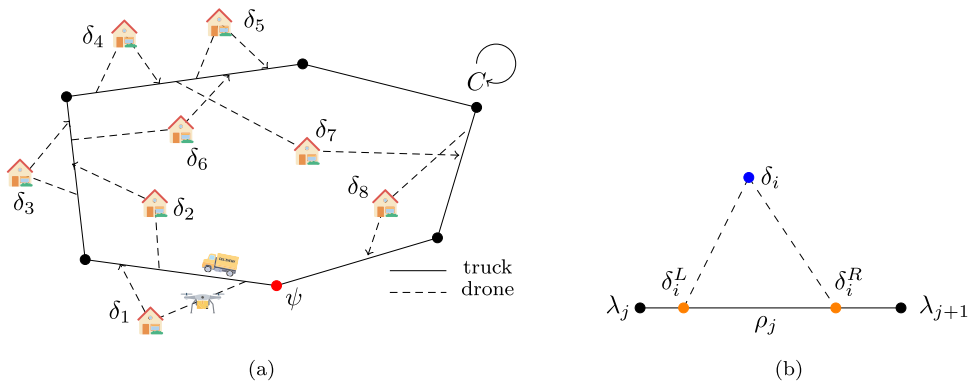



**Fig. 1.** An example delivery area $A$ with 6 roads and 8 customers $\delta_i$ to serve. The depot $\psi$ is at $(0, 0)$; The truck's route is the solid line, while the drones' routes are the dashed lines (a); A road with a delivery $\delta_i$ to do: the launch $\delta_i^L$ and rendezvous $\delta_i^R$ points are highlighted (b).

- The route of the truck is already pre-planned and it is given. Therefore, the truck cannot dynamically make detours or modifications.
- Each drone has only a single battery. Although a drone can perform multiple single deliveries with the same battery, it cannot swap or recharge its battery for additional deliveries.

### 3.2. System and delivery model

A drone's delivery is done by planning a sub-flight that passes through three points. Specifically, the drones take off (with packages) from the truck which continues to drive in the city, deliver packages to the customers, and return to the rendezvous locations with the truck again.

Let $A$ be the 2-D *delivery area* representing our last-mile delivery (application) scenario. Let $\psi \in A$ be the *depot* from where the deliveries start. The position of the depot is located at $(x_\psi, y_\psi)$, assumed to be at the origin of the Cartesian coordinate system in $(0, 0)$. Such a delivery area also comprises roads and customers' positions. At the depot, a *truck* is in charge of transporting a fleet of $m$ *drones* $d_1, \ldots, d_m$ with the same capabilities used for deliveries in $A$. The truck *does not* perform deliveries; it only carries the drones within $A$. Let $\rho_j$ be a straight road delimited by two endpoints $\lambda_j$ and $\lambda_{j+1}$, where $j = 0, \ldots, r$ and $\{\lambda_1, \ldots, \lambda_r\} \subset A$. The truck leaves $\psi$ visiting $\lambda_1$ along $\rho_0 = \overline{\psi\lambda_1}$, then $\lambda_2$ along $\rho_1 = \overline{\lambda_1\lambda_2}$, and so on, up to $\lambda_r$, and eventually going back to $\psi$. These segments make a polygon (*cycle*) $C$ formed by the sequence $\lambda_0, \lambda_1, \ldots, \lambda_r, \lambda_{r+1}$ such that $\lambda_0 = \lambda_{r+1} = \psi$. Let $D = \{\delta_1, \ldots, \delta_n\} \subset A$ be the set of $n$ distinct points or locations to be served (i.e., the *customers*) by the drones. Each customer's (delivery) point $\delta_i$ has a pair of coordinates $(x_{\delta_i}, y_{\delta_i}) \in A$, for $i = 1, \ldots, n$. Fig. 1(a) illustrates the delivery area $A$ with roads and customers.

For each customer's location $\delta_i$, let $\delta_i^L$ and $\delta_i^R$ be respectively the *launch point* and *rendezvous point* of the drone on the truck's route. For any delivery $\delta_i$, both the $\delta_i^L$ and $\delta_i^R$ are already known and given in input. In the example in Fig. 1(b), the truck travels from $\lambda_j$ to $\delta_i^L$ carrying the drone, which in turn takes off at $\delta_i^L$ flying towards $\delta_i$ delivering the package, and finally continues flying towards $\delta_i^R$. In the meanwhile, the truck continues its route reaching other endpoints. When both the truck and the drone arrive at $\delta_i^R$, the former gathers again the drone and they continue to travel up to point $\lambda_{j+1}$. Note that $\delta_i^L$ and $\delta_i^R$ can lie on different roads. In general, $\delta_i^L$ lies on road $\rho_j$ while $\delta_i^R$ lies on road $\rho_z$, where $0 \leq j \leq z \leq r$ (see Fig. 1(a)). Note that, since the truck's route is predefined, it is forbidden for the truck to travel back and forth for picking up the drones. This is our most important assumption which impacts the problem definition and depends on the fact that the truck is, in some sense, an opportunistic means of transportation for the drones. The truck has its own route due to other assigned tasks or because it is moving on a track, as it is common for emerging means of transportation with low greenhouse gas emissions. In any case, the truck cannot change its route.

Let $w_i \geq 0$ be the *energy cost* (or *weight*) for a drone in terms of energy spent to perform a single delivery $\delta_i$ flying to/from the truck. Let $B \geq 0$ be the drone's *energy budget* in terms of battery capacity which limits the total number of feasible deliveries. A drone can perform multiple individual deliveries with a single battery of initial capacity $B$. Each delivery can be performed if it requires less energy than the residual energy budget. In other words, if the drone's current residual energy is not enough for additional flights, it is not possible to perform further deliveries unless a new battery is swapped. We do not consider the possibility to exchange the battery. We assume that drones have to rely only on one battery charge. We remark that, in the multiple drones case, each drone has its own battery of capacity $B$.

Let $p_i \geq 0$ be the *reward* for executing a delivery $\delta_i$. The reward characterizes *premium users* as having higher priority than *regular users*. For instance, delivery companies offer different subscriptions according to the following rule: "*The more you pay, the faster you receive your parcels*". In our context, the higher the priority of delivery $\delta_i$, the larger the reward value $p_i$. Hence, to satisfy the premium users, the scheduling for the drones needs to prioritize the deliveries, guaranteeing first the ones belonging to the premium users (because they have more reward) and second the ones from regular users.

**Table 1**
Symbol description.

| Symbol | Description |
|---|---|
| $A$ | 2-D delivery area |
| $\psi$ | Depot from where the deliveries start |
| $m$ | Number of drones |
| $d_1, \ldots, d_m$ | ID of each drone |
| $\rho_j$ | Straight road delimited by two endpoints |
| $\lambda_j$ | Endpoint of a road |
| $C$ | Cycle of the truck |
| $n$ | Number of deliveries |
| $\delta_i$ | A delivery |
| $(x_{\delta_i}, y_{\delta_i})$ | Location of a delivery |
| $D = \{\delta_1, \ldots, \delta_n\}$ | Set of deliveries |
| $\delta_i^L$ | Launch point of the drone on the truck's route for $\delta_i$ |
| $\delta_i^R$ | Rendezvous point of the drone on the truck's route for $\delta_i$ |
| $w_i$ | Energy cost for a drone to perform delivery $\delta_i$ |
| $B$ | Energy budget for a drone for all the deliveries |
| $p_i$ | Reward for executing a delivery $\delta_i$ |
| $I_i = [\delta_i^L, \delta_i^R]$ | Delivery interval of a delivery $\delta_i$ |
| $I = \{I_1, \ldots, I_n\}$ | Interval set associated with the deliveries |
| $\mathcal{C}(S)$ | Energy cost function for a given subset of deliveries $S \subseteq I$ |
| $\mathcal{P}(S)$ | Reward function for a given subset of deliveries $S \subseteq I$ |

Deliveries are characterized by *intervals* along the truck's route. For any delivery $\delta_i$, the launch and rendezvous points $\delta_i^L$ and $\delta_i^R$ define the drone's *delivery interval* $I_i = [\delta_i^L, \delta_i^R]$. We assume that the truck visits first $\delta_i^L$ and then $\delta_i^R$, so $\delta_i^L \preccurlyeq \delta_i^R$. Let $I = \{I_1, \ldots, I_n\}$ be the *interval set* associated with the deliveries, where $1 \leq i \leq n$. Two intervals $I_i$ and $I_j$ are said to be *compatible* if their intersection is empty, i.e., $I_i \cap I_j = \varnothing$, for $i \neq j$; otherwise, the two intervals are in *conflict*. In other words, $I_i$ and $I_j$ are compatible if $\delta_i^R \preccurlyeq \delta_j^L$ or $\delta_j^R \preccurlyeq \delta_i^L$. This is reasonable with the fact that if a drone is still performing a delivery, i.e., it is still in flight, it cannot start a new delivery without first reaching the truck. A subset $S \subseteq I$ is said to be *compatible* if $I_i \cap I_j = \varnothing$ for any pair $I_i, I_j \in S$. In other words, a drone can perform any subset of such deliveries as long as it has enough battery. Precisely, a given compatible $S \subseteq I$ is *feasible* if the energy cost $\mathcal{C}(S) = \sum_{I_i \in S} w_i \leq B$ (the energy budget). The *reward* of a feasible set $S$ is $\mathcal{P}(S) = \sum_{I_i \in S} p_i$.

Recall that $m$ is the number of drones. Two feasible subsets $S_p, S_q \subseteq I$ can be *assigned* to two drones $d_p$ and $d_q$, with $1 \leq p \neq q \leq m$, if $S_p \cap S_q = \varnothing$. Assuming that $S = \{S_1, \ldots, S_m\}$ consists of $m$ feasible sets assigned to the drones $\{d_1, \ldots, d_m\}$, the *overall reward* $\mathcal{P}(S)$ is defined as the sum of the reward of each drone, i.e., $\mathcal{P}(S) = \sum_{j=1}^{m} \sum_{I_i \in S_j} p_i$.

Table 1 reports the adopted terminology in this paper.

### 3.3. Problem definition

Let us now formally define the delivery scheduling problem.

**Problem 1** (*Scheduling Conflictual Deliveries Problem* (SCDP))**.** Let $\delta_1, \ldots, \delta_n$ be the set of $n$ deliveries, $m$ the number of drones, and $B$ the drone's battery budget. The objective of SCDP is to find a family $S^* = \{S_1^*, \ldots, S_m^*\} \subseteq I$ of $m$ feasible subsets with $S_p^* \cap S_q^* = \varnothing$, for $1 \leq p \neq q \leq m$, such that the overall reward $\mathcal{P}(S)$ is maximized. Specifically,

$$S^* = \underset{S=\{S_1,\ldots,S_m\} \subseteq I}{\arg\max} \ \mathcal{P}(S) \qquad \text{such that} \quad \mathcal{C}(S_i) \leq B \ \forall i = 1, \ldots, m$$

In the following we show the *NP*-hardness of SCDP, even for the single drone case.

An alternative way to define the set of deliveries and constraints is to observe that the set of intervals $I$ can be described as an *interval graph*. In SCDP, the set $D$ determines the set of intervals $I$ that can be pairwise compatible or in conflict with respect to their launch and rendezvous points. Given an instance of SCDP, we can visualize the intervals and their compatibility along a temporal line. Fig. 2(a) shows the intervals corresponding to Fig. 1(a), in which $I_2$ is in conflict with both $I_1$ and $I_3$, whereas $I_1$ and $I_3$ are compatible.

A simple way to figure out if two intervals are in conflict or not is by considering the interval graph representation. Formally, in the *undirected interval graph*, $G = (V, E)$, the vertex-set $V = I$, where each vertex uniquely corresponds to an interval $I_i \in I$, for $i = 1, \ldots, n$; and an edge $(I_i, I_j) \in E$ indicates that the deliveries $\delta_i$ and $\delta_j$ are not compatible, implying $I_i \cap I_j \neq \varnothing$. For example, in Fig. 2(a), $I_1$ and $I_3$ can be executed without conflicts, i.e., $(I_1, I_3) \notin E$; while $I_2$ is not compatible with both $I_1$ and $I_3$, implying $(I_2, I_1), (I_2, I_3) \in E$. Note that vertex $I_6$ has four conflicts because its degree is 4.

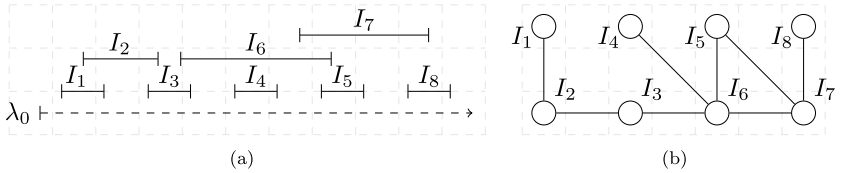



**Fig. 2.** Delivery intervals corresponding to Fig. 1(a)(a); Interval graph $G$ for the example in Fig. 1(a)(b).

In the following we demonstrate that SCDP is an *NP*-hard problem by showing that the classic 0–1 Knapsack Problem (KP) is a special case of SCDP when intervals are conflict-free.

**Theorem 1.** SCDP *is NP-hard.*

**Proof.** We start proving that the simpler scenario of SCDP with a single drone is *NP*-hard. Our approach is by reduction from KP, known to be *NP*-hard [27] and defined as follows. Given a set $X = \{1, \ldots, n\}$ of $n$ items, where each item $i$ is associated with a cost $w_i$ and reward $p_i$ , and a knapsack of capacity $c$, the KP is to find a subset of $X$ that maximizes the sum of the rewards, satisfying the capacity constraint. Given an instance of KP, we translate it as an instance of SCDP as follows: we first set the energy budget of SCDP equal to the capacity constraint of the knapsack, i.e., $B = c$. Then, we create a set of deliveries $D$ starting from the set of items $X$ as follows: for each item $i \in X$, create a delivery $\delta_i \in D$ with equal cost and reward. We create deliveries with cost $w_i$ and reward $p_i$. Finally, we create each interval $I_i = [\delta_i^L, \delta_i^R]$ with constant span $s > 0$, defining its launch and rendezvous point as $\delta_i^L = \delta_{i-1}^R + s$ and $\delta_i^R = \delta_i^L + s$, respectively, assuming that $I_1 = [0, s]$. Hence, we can observe that the deliveries are pairwise compatible in the corresponding SCDP instance. Thus, a solution for KP is a solution for SCDP and vice versa. This reduction takes polynomial time. Now, since the case with a single drone is *NP*-hard, also the general multiple drones case of SCDP is *NP*-hard. □

### 3.4. The optimal algorithm

For optimally solving SCDP in the general case with $m$ drones, we can use an ILP formulation. We enumerate the deliveries as $\mathcal{N} = \{1, \ldots, n\}$, and drones as $\mathcal{M} = \{1, \ldots, m\}$. Let $z_{ij} \in \{0, 1\}$ be a decision variable that is 1 if the delivery $j \in \mathcal{N}$ is accomplished by the drone $i \in \mathcal{M}$; otherwise it is 0. Therefore, the ILP formulation is given by:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_j z_{ij} \tag{1}$$

subject to:

$$\sum_{j=1}^{n} w_j z_{ij} \leq B, \qquad \forall i \in \mathcal{M} \tag{2}$$

$$\sum_{i=1}^{m} z_{ij} \leq 1, \qquad \forall j \in \mathcal{N} \tag{3}$$

$$z_{ij} + z_{ik} \leq 1, \qquad \forall i \in \mathcal{M}; \forall j, k \in \mathcal{N} s.t.\ I_j \cap I_k \neq \varnothing \tag{4}$$

$$z_{ij} \in \{0, 1\}, \qquad \forall i \in \mathcal{M}, \forall j \in \mathcal{N} \tag{5}$$

The objective function (1) aims to maximize the overall reward using a fleet of $m$ drones. About the constraints, (2) states that each drone has a capacity budget of $B$ in terms of energy; (3) forces each delivery to be executed by at most one drone; and (4) avoids the fact that two incompatible deliveries are performed by the same drone. (5) simply sets the domain of the decision variable.

In the next following, we propose algorithms for solving SCDP in scenarios involving single and multiple drones. From now on, we will interchangeably use the terms "delivery" and "interval".

## 4. Algorithms for SCDP with a single drone

This section proposes three algorithms for SCDP with a single drone (with -S suffix) – an optimal knapsack-based algorithm (KNA-S), and two approximation algorithms, i.e., a coloring-based (COL-S) and a bin packing-based (BIN-S) whose computational time complexity is polynomial. KNA-S can be performed only when $B \in \mathbb{N}$, and it is suitable only for small values of $B$. For larger values of $B$, we devise COL-S and BIN-S. Namely, COL-S first partitions the deliveries into sets of conflict-free deliveries and then extract from them the subset of deliveries with the maximum rewards feasible with the available energy budget; whereas BIN-S first extracts the most profitable set of conflict-free deliveries and then partitions them based on the energy availability.

### 4.1. The KNA-S optimal algorithm

We propose a pseudo-polynomial time algorithm based on dynamic programming that optimally solves SCDP with a single drone, called KNA-S, requiring $\mathcal{O}(n \log n + nB)$ time and $\mathcal{O}(nB)$ space. We remark that, in this case, due to the dynamic programming approach, we require that both weights $w_i$ and budget $B$ must be integers to let the algorithm return the optimal solution.

Before proceeding, let $\sigma(i)$ be the largest index $1 \leq j \leq i-1$ such that $I_i \cap I_j = \varnothing$, and $\delta(i) = 0$ if no interval $j < i$ is disjoint from $i$. We prove the following property: if the set of intervals is sorted in non-decreasing order of the rendezvous points along the truck's route, i.e., $\delta_i^R \preccurlyeq \delta_{i+1}^R$, then the intervals intersecting with any given interval $I_i$ are adjacent in the ordering. Our proposed dynamic programming solution exploits this property as follows: for each interval $I_i$, we can find the largest index $\sigma(i)$ that precedes $i$ without intersecting with it.

**Lemma 1.** *Let the set of intervals $I$ be sorted in the non-decreasing order of rendezvous points, i.e., $\delta_1^R \preccurlyeq \ldots \preccurlyeq \delta_n^R$. Given an interval $I_i$, there exists an index $\sigma(i) < i$ such that $I_k \cap I_i = \varnothing \quad \forall k \in 1, \ldots, \sigma(i)-1$ and $I_k \cap I_i \neq \varnothing \quad \forall k \in \sigma(i), \ldots, i$.*

**Proof** (*By Contradiction*)**.** Assume there exist indices $\ell < \kappa < \sigma(i)$ such that the intervals between $\kappa$ and $\sigma(i)$ do not intersect with $I_i$ but $I_\ell \cap I_i \neq \varnothing$. That is, $I_k \cap I_i = \varnothing \quad \forall k \in \kappa, \ldots, \sigma(i)-1$ and $I_\ell \cap I_i \neq \varnothing$. This implies that $\delta_\ell^R \succ \delta_i^L$ since the intervals intersect with each other, and $\delta_k^R \prec \delta_i^L$ for all $k \in \kappa, \ldots, \sigma(i)-1$. So, $\delta_\ell^R \succ \delta_k^R$, which contradicts that $I$ is sorted in non-decreasing order of rendezvous points. In other words, $\ell < \kappa < \sigma(i)$ implies $\delta_\ell^R \prec \delta_\kappa^R \prec \delta_{\sigma(i)}^R$. □

Let us now introduce our novel KNA-S algorithm for SCDP based on dynamic programming. During the initialization phase, a table $M$ of size $n \times B$ is created and the set of intervals is sorted in non-decreasing order of rendezvous points. To compute the value in each cell of $M$, we define a function $g(i, b)$ as the maximum reward achievable by considering the first $i$ sequences and budget $b$. Hence, we set $M[i, b] = g(i, b)$ for $0 \leq i \leq n$ and $0 \leq b \leq B$. To compute $g(i, b)$ using dynamic programming, for each set of sequences $\{0, \ldots, i\}$ and for each budget $b = 0, 1, \ldots, B$, we first set $M[0, b] = M[i, 0] = 0$ for each $i, b$. Then, for each subsequent row and column, $M[i, b]$ is the maximum between the solution $g(i-1, b)$ with an equal budget that does not consider element $i$, and the reward of element $i$ plus the optimal solution compatible with $i$. For the latter, it is necessary to find a solution that satisfies both the budget constraint and the possible intersections with the newly added element $i$. Precisely,

$$g(i, b) = \begin{cases} 0 & \text{if } i = 0 \text{ or } b = 0 \\ g(i-1, b) & \text{if } w_i > b \\ \max\{g(i-1, b), p_i + g(\sigma(i), b - w_i)\} & \text{otherwise} \end{cases}$$

Note that the above definition exploits the optimality of the sub-problems $g(i-1, b)$ and $g(i, b)$. To efficiently implement the $\sigma$ function a preprocessing phase is required. First, let us order the set of launch and rendezvous points associated with the intervals in non-decreasing order; then consider an empty stack and a pointer to the beginning of the ordered set. If the element indexed by the pointer is a rendezvous point, insert the associated interval on the stack and move the pointer by one position. Otherwise (launch point), associate the interval at the top of the stack as the predecessor $\sigma(i)$ of the interval $i$ associated with the launch point indexed by the pointer and move the pointer one position. If the stack is empty, $\sigma(i) = \varnothing$ is associated. Finally, each interval has an associated predecessor. This procedure requires $\mathcal{O}(n \log n + n)$ to sort the launch and rendezvous points and to associate the predecessors, while it takes $\mathcal{O}(n)$ space.

**Theorem 2.** KNA-S *optimally solves* SCDP *with a single drone in $\mathcal{O}(n \log n + nB)$ time.*

**Proof.** For the correctness of optimality, we use induction. The base case $i = 0$ is trivial since there is no interval to include in the solution. Thus, $M[0, b] = g(0, b) = 0$.

*Inductive Step*: When computing $M[i, b]$, by the induction hypothesis, $M[i-\ell_1, b-\ell_2]$ for any $1 \leq \ell_1 \leq i$ and $0 \leq \ell_2 \leq b$ are already computed correctly. There are two possible cases. If $w_i > b$, then $M[i, b] = M[i-1, b] = g(i-1, b)$ which was optimum by induction hypothesis; thus $M[i, b]$ is optimum since it is not possible to add interval $I_i$. Otherwise, if $w_i \leq b$ we have $M[i, b] = \max\{g(i-1, b), p_i + g(\sigma(i), b - w_i)\}$. Moreover, as before, $g(i-1, b)$ is optimum by induction. As proven in Lemma 1, the set of sequences $I_1, \ldots, I_{\sigma(i)}$ contains all the sequences that do not intersect with $I_i$. Hence, by induction, $g(\sigma(i), b - w_i)$ contains the optimal solution for the set of intervals $I_1, \ldots, I_{i-1}$ that is also feasible with $I_i$. The time complexity depends on the preprocessing which is $\mathcal{O}(n \log n)$ to compute the predecessor, and on the running time $\mathcal{O}(nB)$ since the size $n \times B$ of table $M$. This permits us to calculate in constant time function $\sigma(i)$ for each interval, and hence we can fill each cell of table $M$ in constant time. □

Given that the time complexity of the optimal KNA-S algorithm is pseudo-polynomial time in the budget size $B$, in the following, we search for faster solutions that sacrifice the optimality.

### 4.2. The COL-S approximation algorithm

The COL-S approximately solves SCDP with a single drone, requiring $\mathcal{O}(n \log n + h(n))$ time and $\mathcal{O}(n)$ space, where $h(n)$ is the time required by a subroutine used in it.

The idea of COL-S is that we first adopt an optimal polynomial-time vertex graph coloring algorithm for interval graphs to divide the set of intervals into several subsets based on the minimum coloring of $I$ [28]. Each color represents a subset of deliveries without conflicts that can be assigned to a single drone. Then, we find a solution for SCDP for each subset, that is, we select the most profitable subset of deliveries in each subset of independent deliveries subject to the energy budget constraint. Finally, we return the solution with the maximum reward among all the colors. The pseudo-code of COL-S is given in Algorithm 1.

First, we find an optimal vertex-coloring for the graph induced by $I$ dividing the set of intervals $\{I_1, \ldots, I_n\}$ into $\chi$ distinct subsets. Recall that, a vertex coloring is an assignment of labels or colors to each vertex of a graph such that no edge connects two identically colored vertices. The minimum number of colors for a feasible vertex coloring in a graph $G$ is called the *chromatic number* of $G$, denoted $\chi(G)$. Moreover, a $k$-coloring of a graph $G$ is a feasible vertex coloring that assigns the integers $1, \ldots, k$ (the colors) to the vertices of $G$. Finding the minimum number $\chi$ of colors for proper coloring in arbitrary graphs is an *NP*-hard problem, but it is solvable in polynomial time for the special case of interval graphs [28].

**Algorithm 1:** The COL-S Algorithm

1 $C_1, \ldots, C_\chi \leftarrow$ `coloring`$(I)$;
2 **for** $i \in 1, \ldots, \chi$ **do**
3 $\quad S_i \leftarrow \max_{S \in C_i : \mathcal{C}(S) \leq B} \mathcal{P}(S)$
4 **return** $SOL \leftarrow \max_{i \in 1, \ldots, \chi} \mathcal{P}(S_i)$

To implement `coloring` (Line 1, Algorithm 1), let us first order in a list the set of launch and rendezvous points associated with the intervals in non-decreasing order. Then consider a pointer $i$ initialized to the first point of the ordered set of launch and rendezvous points, and a min-heap data structure that stores the colors available. The min-heap is initialized with only one color, which is the first color made available. The coloring algorithm proceeds by moving the pointer on the sorted list. If the element pointed by the pointer is a launch point, the available color in the heap with the minimum index is associated with interval $i$ that starts at the pointed launch point. If the element pointed by the pointer is a rendezvous point, the color used for the interval $i$ that finishes is inserted in the heap and made available in the min-heap for further intervals. Finally, move the pointer of one position. After extracting, if the heap is empty, insert a new color (i.e., a new available integer not used until now). Note that the overall time complexity is $O(n \log n)$ where $n$ is the number of intervals (i.e., deliveries) to be served.

The `coloring` procedure (Line 1) takes in input the set of intervals $I$, and returns as output $\chi$ subsets $C_1, \ldots, C_\chi$ of deliveries, where $\chi$ is the chromatic number of the graph[1] associated with the intervals $I$. That is, the coloring partitions $I$ into $\chi$ subsets of conflict-free deliveries, one for each color. After `coloring` is performed, for each subset $C_i$, find a subset $S_i \subseteq C_i$ with maximum reward such that $\mathcal{C}(S_i) \leq B$ (Line 3). Note that, subsets $S_i$ for each $i$ are feasible solutions for SCDP. Finally, we return the solution $SOL$ with the maximum reward among all the subsets $S_i$ (Line 4).

It is worth noting that finding an optimal set $S_i$ is equivalent to solving a Knapsack problem (KP) with budget $B$ on the elements $C_i$, which is an *NP*-hard problem, but which can be approximated in polynomial time.

**Theorem 3.** COL-S *provides a solution for* SCDP *for a single drone with an approximation ratio of* $\frac{\alpha}{\chi}$*, where* $\chi$ *denotes the chromatic number of the interval graph induced from the deliveries in I, and* $\alpha$ *is the approximation ratio of an algorithm that maximizes* KP.

**Proof.** Suppose we have a proper $\chi$-coloring of the set $I$ that partitions the interval sequences into $\chi$ color classes, namely $C_1, \ldots, C_\chi$. Let $S_i$ denote the set with maximum reward in color class $C_i$, (Line 3, Algorithm 1). Then, let $SOL = \max_{i \in 1, \ldots, \chi} \mathcal{P}(S_i)$ be the set with maximum reward among all sets $S_i$ such that $\mathcal{C}(S_i) \leq B$. Let $OPT$ be an optimal solution for the SCDP on the interval set $I$. Clearly, the intervals selected by $OPT$ are partitioned among $\chi$ colors, i.e., $OPT = \cup_{1 \leq i \leq \chi} (OPT \cap C_i)$. We first note that $\mathcal{P}(OPT \cap C_i) \leq \mathcal{P}(S_i)$ for $i \in 1, \ldots, \chi$. It follows directly, if either $OPT \cap C_i = S_i$ or $OPT \cap C_i = \varnothing$. Otherwise we could have $(OPT \cap C_i) \cap S_i \neq \varnothing$. In this case, $\mathcal{P}(OPT \cap C_i) > \mathcal{P}(S_i)$ will contradict that $S_i$ is the subset of $C_i$ with maximum reward since $S_i = \max_{S \subseteq C_i} \mathcal{P}(S)$.

Recall that finding an optimal set $S_i$ is equivalent to solve the KP for elements $C_i$ with budget $B$, which is an *NP*-hard problem. For each set $C_i$, let us consider two solutions $S_i$ and $S_i^*$ for the problem of maximizing the reward in $C_i$, where

[1] Recall that the graph has a vertex for each delivery, and an edge between any two intervals that overlap.

$S_i^*$ is the optimal solution to this problem and $S_i$ is an $\alpha$-approximation to $S_i^*$. Thus,

$$\mathcal{P}(SOL) = \frac{1}{\chi}\sum_{i=1}^{\chi}\mathcal{P}(SOL) \geq \frac{1}{\chi}\sum_{i=1}^{\chi}\mathcal{P}(S_i) \geq \frac{\alpha}{\chi}\sum_{i=1}^{\chi}\mathcal{P}(S_i^*) \geq \frac{\alpha}{\chi}\sum_{i=1}^{\chi}\mathcal{P}(OPT \cap C_i) \geq \frac{\alpha}{\chi}\mathcal{P}(OPT). \quad \square \quad \square$$

Once the optimal coloring has been computed, there are $C_1, \ldots, C_\chi$ partitions where $\chi$ denotes the minimum number of subsets without conflicts. Then, the next phase is to determine a subset $S_i$ for each $C_i$ such that the reward is maximized with the constraint of the maximum allowed budget (Line 3). In other words, we need to solve the KP on each subset $C_i$ whose size is $n_i = |S_i|$, and $\sum_{i=1}^{\chi} n_i = n$. According to Theorem 3, the Col-S algorithm can solve SCDP (with a single drone) guaranteeing a lower bound of $\frac{\alpha}{\chi}$ with respect to the optimal solution, where the parameter $\alpha$ depends on how we determine the different subsets $S_i$.

A few words to help analyze $\alpha$ and $h(n)$ in the time complexity. An approach is to rely on a fully polynomial time approximation scheme (FPTAS) with $\alpha = 1 - \epsilon$, where $0 < \epsilon < 1$, requiring a computational time of $\tilde{\mathcal{O}}(n_i + (\frac{1}{\epsilon})^{2.5})$ [29], where the $\tilde{\mathcal{O}}$ notation hides poly-logarithmic factors in $n_i$ and $\frac{1}{\epsilon}$ [29]. So, a fast greedy strategy for the *fractional knapsack* problem can be exploited which guarantees $\alpha = 0.5$ [30]. Finally, to maximize $\alpha$ keeping polynomial the time complexity of the algorithm, it is possible to exploit the submodularity property[2] thus obtaining a solution that guarantees $\alpha = 1 - e^{-1} \approx 0.63$ [32] taking $\mathcal{O}(n_i^5)$ time for each $C_i$ and overall $\mathcal{O}(n^5)$ time. As a last remark we note that, in the case all the deliveries require the same cost, i.e., $w_j = \gamma \in \mathbb{N}_{\geq 0}$, for $1 \leq j \leq n_i$, $\alpha = 1$ because it is simply required to sort the $n_i$ elements of $C_i$ in non-decreasing order by the reward and then pick the first $\lfloor B/\gamma \rfloor$ elements. Overall this takes $\mathcal{O}(n)$ time using the selection algorithm to locate the item of rank $\lfloor B/\gamma \rfloor$ in each $C_i$ in $\mathcal{O}(n_i)$, for $1 \leq i \leq n$ [33].

### 4.3. The Bin-S approximation algorithm

The Bin-S approximately solves SCDP with a single drone, requiring $\mathcal{O}(n \log n + h(n))$ time and $\mathcal{O}(n)$ space, where $h(n)$ is the time required by a subroutine used in it.

The algorithm relies on the Bin Packing Problem (BPP). Recall that, the BPP is an optimization problem in which items of different sizes must be packed into a finite number of bins or containers, each of a fixed given capacity, in a way that the number of used bins is minimized. Both Col-S and Bin-S search for conflict-free subsets of intervals but use different strategies: Bin-S selects a subset of independent intervals that maximize the reward and, on this, refines the intervals' selection to satisfy the budget constraint. In more detail, we optimally compute the *maximum reward subset* of compatible intervals of $I$ in polynomial time by solving via dynamic programming the Weighted Interval Scheduling Problem (WISP). Then, we optimize the interval selection returned by WISP by grouping the solution in bins of size (energy cost) at most $B$ via a greedy implementation of the BPP. Finally, Bin-S outputs the bin with the maximum reward between all the bins. The pseudo-code of Bin-S is given in Algorithm 2.

**Algorithm 2:** The Bin-S Algorithm

```
1 SOL_max ← weighted-interval-scheduling(I);
2 S_1, ..., S_η ← bin-packing(SOL_max)
3 return SOL ← max_{i∈1,...,η} P(S_i)
```

Initially, we find the subset with maximum reward of compatible deliveries $SOL_{\max}$ by executing the optimal algorithm for the WISP [34] on the set $I$ in input (Line 1, Algorithm 2). Then, by invoking `bin-packing` on $SOL_{\max}$, we get $\eta$ bins $S_1, \ldots, S_\eta$ such that $\mathcal{C}(S_i) \leq B$ (Line 2), where $\eta$ is the optimum (minimum) number of required bins. On each bin, deliveries are compatible with each other. Eventually, the bin with maximum reward, $SOL$, is returned in output (Line 3).

Note that finding the optimal division of deliveries into $S_1, \ldots, S_\eta$ bins is equivalent to solving BPP with budget $B$ on the subset of deliveries $SOL_{\max}$. However, since BPP is strongly *NP*-hard, we only consider polynomial time $\beta$-approximated solutions for it, which in turn also affects the approximation ratio of Bin-S. Recall that an approximated version of `bin-packing` returns a solution with a number of bins that is no more than $\beta \geq 1$ times the optimal number $\eta$ of bins.

**Theorem 4.** Bin-S *provides a solution for* SCDP *for a single drone with an approximation ratio of* $\frac{1}{\lceil \beta\eta \rceil}$*, where* $\eta$ *denotes the optimal number of bins, and* $\beta$ *is the approximation ratio of an algorithm that minimizes* BPP.

**Proof.** Let $SOL_{\max}$ be the subset of compatible intervals of $I$ with maximum reward $P_{\max}$, and $OPT$ be the optimal solution for SCDP. Suppose we have an optimal solution for BPP on $SOL_{\max}$ which partitions $I$ into $\eta$ bins $S_1, \ldots, S_\eta$ such that $\mathcal{C}(S_i) \leq B$. By the *pigeonhole principle* [35] there exists at least one $\hat{i}$ such that $\mathcal{P}(S_{\hat{i}}) \geq \frac{P_{\max}}{\eta}$. Let $S_{\hat{i}}$ be the solution $SOL_B$ of

[2] Submodularity [31] Given a finite set $X = \{\xi_1, \ldots, \xi_n\}$, a set function $\mathcal{F} : 2^X \to \mathbb{R}$ is submodular if for any $S \subseteq T \subseteq X$ and $\xi \in X \setminus T$, $\mathcal{F}(S \cup \xi) - \mathcal{F}(S) \geq \mathcal{F}(T \cup \xi) - \mathcal{F}(T)$.

BIN-S for SCDP, and recall that $P_{\max} \geq OPT$ by definition since $P_{\max}$ does not consider the budget constraint. Since we rely on a $\beta$-approximation algorithm for BPP, the returned number of bins is $\lceil \beta\eta \rceil$. Hence:

$$\mathcal{P}(SOL_B) \geq \frac{P_{\max}}{\eta} \geq \frac{P_{\max}}{\lceil \beta\eta \rceil} \geq \frac{OPT}{\lceil \beta\eta \rceil} \implies \frac{\mathcal{P}(SOL_B)}{\mathcal{P}(OPT)} \geq \frac{1}{\lceil \beta\eta \rceil}. \quad \square \ \square$$

Notice that the guaranteed approximation bound of BIN-S depends on the number of bins returned by the algorithm used to solve BPP, which in turn depends on $\beta$. Trivially, $\beta = 1$ if we perform an optimal and computationally expensive algorithm for BPP. Another approach is to rely on the approximation algorithm devised in [36] that gives $\beta \approx 1.6$ taking $\mathcal{O}(n \log n)$ time but with a potential unbounded space complexity without any further assumption. The Best-Fit (BF) algorithm guarantees $\beta = 1.7$ in $\mathcal{O}(n \log n)$ time and $\mathcal{O}(n)$ space [37]. A faster algorithm that provides $\beta = 2$ with both $\mathcal{O}(n)$ time and space, is Next-Fit (NF) [38]. However, in this paper, we prefer BF since it has the best trade-off.

## 5. Algorithms for SCDP with multiple drones

This section proposes three algorithms for SCDP with multiple drones (with -M suffix) – a knapsack-based approximation algorithm (KNA-M), a bin packing-based approximation algorithm (BIN-M), and a coloring-based heuristic algorithm (COL-M). Note that, these algorithms have been devised and adapted from the original single-case drone algorithms in Section 4.

### 5.1. The KNA-M approximation algorithm

Based on dynamic programming, the KNA-M algorithm solves SCDP with multiple drones, requiring $\mathcal{O}(m(n \log n + nB))$ time and $\mathcal{O}(nB)$ space.

The idea behind KNA-M is that we sequentially run an optimal strategy (i.e., KNA-S) on the current residual intervals for each drone. Then, we find a global solution for SCDP which comprises all the previously computed solutions. The pseudo-code of KNA-M is given in Algorithm 3.

**Algorithm 3:** The KNA-M Algorithm

```
1 Î ← I;
2 for i ∈ 1, ..., m do
3 |  S_i ← KNA-S(Î), Î ← Î \ S_i;
4 return SOL ← {S_1, ..., S_m}
```

First, the set $\hat{I}$ characterizing the current residual intervals is initialized with all the intervals $I$ (Line 1, Algorithm 3). Then, recalling that there are $m$ drones, we iteratively (Line 2) invoke the optimal solution KNA-S on $\hat{I}$ (Line 3), which contains the intervals (deliveries) that have not been assigned to drones yet. This step makes an intermediate solution $S_i$ at each iteration for the $i$th drone. So, the current set of intervals available for the remaining drones is decreased (Line 3). Finally, the returned solution is simply the union of the previously computed sub-solutions (Line 4).

The guaranteed approximation ratio of KNA-M is proven in Theorem 5.

**Theorem 5.** *The* KNA-M *algorithm provides a solution for* SCDP *for multiple drones with an approximation ratio of* $\frac{1}{m}$*, where* $m$ *denotes the number of drones.*

**Proof.** Let $SOL = \{S_1, \ldots, S_m\}$ be the solution returned by KNA-M and let $OPT = \{O_1, \ldots, O_m\}$ be an optimal solution for the SCDP on $I$. By construction of the solution $SOL$, we have $\mathcal{P}(S_1) \geq \cdots \geq \mathcal{P}(S_m)$. Note that having $\mathcal{P}(S_j) \geq \mathcal{P}(S_i)$, for $i < j$, will contradict that $S_i$ is the subset with maximum reward at step $i$ of algorithm KNA-M. Moreover, since $S_1$ is the optimal solution with one drone, $\mathcal{P}(S_1) \geq \mathcal{P}(O_i)$, $\forall i \in \{1, \ldots, m\}$. Thus,

$$\mathcal{P}(SOL) \geq \mathcal{P}(S_1) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{P}(S_1) \geq \frac{1}{m} \sum_{i=1}^{m} \mathcal{P}(O_i) = \frac{1}{m} \mathcal{P}(OPT). \quad \square \ \square$$

The KNA-M algorithm takes $\mathcal{O}(m(n \log n + nB))$ time, and $\mathcal{O}(nB)$ space.

### 5.2. The BIN-M approximation algorithm

Similarly to KNA-M, BIN-M iterates the BIN-S algorithm $m$ times, every time working on the remaining deliveries to find a solution that approximates SCDP. BIN-M solves SCDP with multiple drones in $\mathcal{O}(m(n \log n + h(n)))$ time and $\mathcal{O}(n)$ space. The pseudo-code of BIN-M is similar to the one already shown in Algorithm 3. The only difference resides in Line 3, where in BIN-M is iteratively invoked BIN-S. The pseudo-code of BIN-M is given in Algorithm 4.

Similarly to Theorem 5, it can be proven:

**Algorithm 4:** The BIN-M Algorithm

$\hat{I} \leftarrow I$;
**for** $i \in 1, \ldots, m$ **do**
  $S_i \leftarrow$ BIN-S($\hat{I}$), $\hat{I} \leftarrow \hat{I} \setminus S_i$;
**return** $SOL \leftarrow \{S_1, \ldots, S_m\}$

**Theorem 6.** *The* BIN-M *algorithm provides a solution for* SCDP *for multiple drones with an approximation ratio of* $\frac{1}{\lceil m\beta\eta_1 \rceil}$, *where m denotes the number of drones,* $\eta_1$ *is the optimal number of bins, and* $\beta$ *is the approximation ratio of an algorithm that solves BPP during the first iteration.*

### 5.3. The COL-M heuristic algorithm

The heuristic algorithm COL-M solves SCDP with multiple drones in $\mathcal{O}(m(n \log n + h(n)))$ time and $\mathcal{O}(n)$ space. The idea behind COL-M is that we sequentially perform a conflict-free coloring using COL-S on the current residual intervals, so we assign a drone for each sub-partition. Then, we find a global solution for SCDP comprising the previously computed solutions. The pseudo-code of COL-M is given in Algorithm 5.

**Algorithm 5:** The COL-M Algorithm

1 $\hat{I} \leftarrow I, m' \leftarrow m, SOL \leftarrow \varnothing$;
2 **while** $\hat{I} \neq \varnothing$ **do**
3  $\{S_1, \ldots, S_\chi\} \leftarrow$ COL-S($\hat{I}$);
4  sort($S_i$) s.t. $\mathcal{P}(S_i) \geq \mathcal{P}(S_{i+1})$;
5  $SOL \leftarrow SOL \cup \{S_1, \ldots, S_{\min\{\chi, m'\}}\}$;
6  $\hat{I} \leftarrow \hat{I} \setminus \bigcup S_i, m' \leftarrow m' - \min\{\chi, m'\}$;
7 **return** $SOL$;

The set $\hat{I}$ and the number $m'$ of available drones are first initialized (Line 1). Next, until $\hat{I}$ is not empty (Line 2), an optimal coloring is performed to generate $S_1, \ldots, S_\chi$ (Line 3), which are sorted in non-decreasing order by the total reward (Line 4). Now, depending on the number of current available groups $\chi$ and drones $m'$, we assign the best $\min\{\chi, m'\}$ groups to drones (Line 5). After that, we update the current solution *SOL* and the number of available drones (Line 6).

## 6. Performance evaluation

In this section, we compare the performances, in terms of obtained reward, of the three proposed heuristics for solving SCDP with a single drone and multiple drones cases.

### 6.1. The settings

In the last-mile delivery area $A$, we assume for the truck a randomly generated trip of 300 km, which is a reasonable distance for a single working day [39]. The locations of the $n = \{25, 50, 75, 100\}$ deliveries are uniformly generated at random in $A$. The truck carries a total of $m = \{1, 3, 5\}$ drones that fly at a constant speed $d_s = 20$ m/s [40]. We set the drone's battery $B = 5$ MJ [40]. We set the drone's payload to $d_p = 5$ kg [40]. With respect to these parameters, the energy cost $w_i$ for performing a delivery is computed according to the energy model presented in [40,41], which depends on the distance to travel, the total mass of the drone plus the payload, and the drone's speed.

About the rewards assigned to each delivery, we randomly generate them as an integer number between [1, 100] according to the Zipf distribution [42] varying the $\theta$ parameter in {0, 0.4, 0.8, 1.0}. When $\theta = 0$, the rewards are uniformly distributed in [1, 100]; when $\theta \geq 0.8$, there are a few rewards with a large probability and many rewards occurring a few times. The energy costs and the delivery intervals are uniformly generated according to the Uniform distribution, assuming four *energy/delivery-interval configurations* $\Sigma_i$ (from $\Sigma_1$ with low variability to $\Sigma_4$ with high variability). In this case, the delivery interval $I_i$ of a given delivery $\delta_i$ is simply the length between $\delta_i^L$ and $\delta_i^R$ on the truck's road. With regard to the configurations, in $\Sigma_1$ the maximum energy cost is 2.5 MJ and the maximum delivery interval is 1.5 km, in $\Sigma_2$ this pair is 5 MJ and 10 km, in $\Sigma_3$ we have 7.5 MJ and 20 km, and finally in $\Sigma_4$ we have 30 MJ and 30 km. Note that, $\Sigma_1$ and $\Sigma_2$ always admit feasible deliveries, while the other two do not. Moreover, $\Sigma_4$ allows very long delivery intervals. Finally, once the delivery intervals are generated, each launch point $\delta_i^L$ is uniformly generated between $\lambda_0$ and $\lambda_{r+1}$ minus the length of the interval $I_i$.

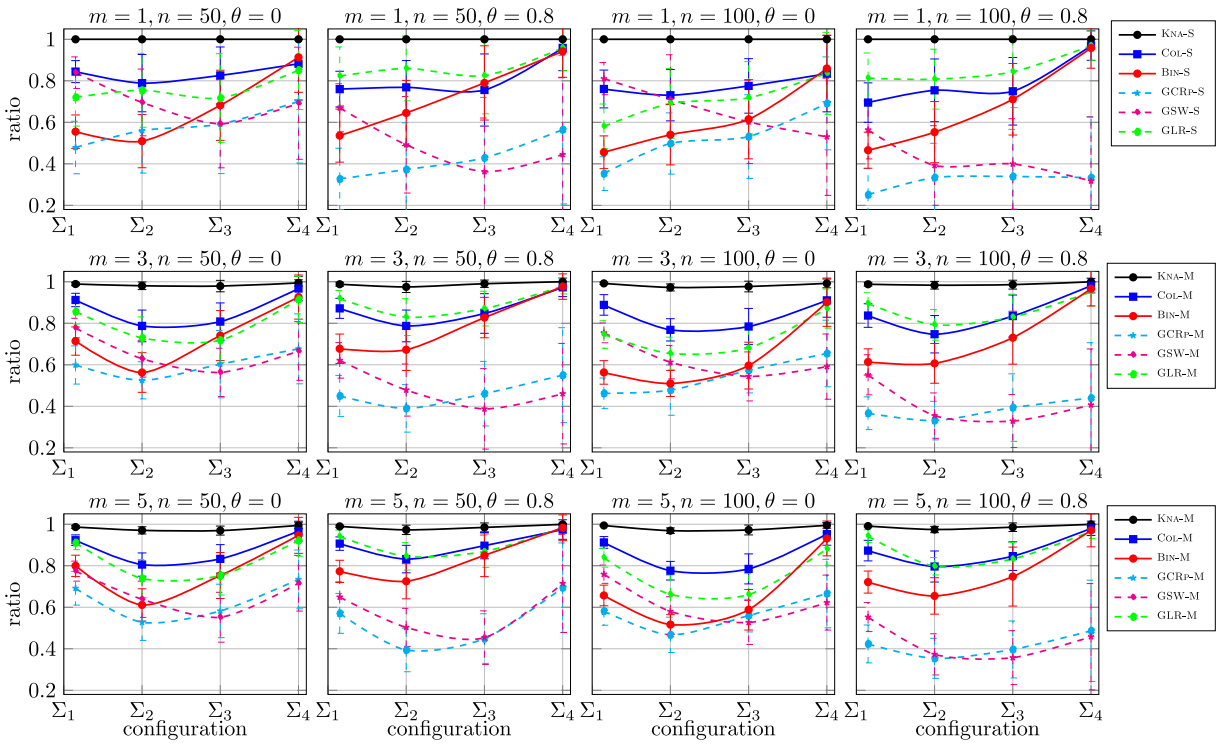



**Fig. 3.** Performance evaluation of our algorithms. The first row compares algorithms with a single drone, while the other two rows compare with multiple drones.

When evaluating the performances of our algorithms presented in Sections 4–5, we compare COL-S and BIN-S with respect to the optimal solution in the single drone scenario provided by KNA-S, and we compare KNA-M, BIN-M, and COL-M with respect to the optimum solutions in the multiple drones scenario obtained from the ILP formulation. Also, as a reference for comparison (as baseline), we propose three greedy heuristic algorithms for single and multiple drone scenarios. Specifically, *Greedy Closest Rendezvous Point* (GCRP) repeatedly selects the compatible interval with the closest rendezvous point $\delta_i^R$, *Greedy Smallest Weight* (GSW) repeatedly selects the compatible interval with the smallest energy cost $w_i$, *Greedy Largest Reward* (GLR) repeatedly selects the compatible interval with the largest reward $p_i$. In the case of multiple drones, these heuristics are repeated on the residual subset of deliveries not yet assigned to the drones.

### 6.2. Experiment results

Fig. 3 illustrates the performances, in terms of collected rewards, of our algorithms on synthetic data. Specifically, in the first row, we present the results for SCDP with $m = 1$ (single drone), whereas in the other two rows we illustrate the results for SCDP with $m = 3$ and $m = 5$ drones, respectively. The $x$-axis reports the four different configurations *energy/interval* $\Sigma_i$, while the $y$-axis shows the ratio between the reward reported by any algorithm and that by the optimum algorithm. Clearly, the ratio is less than or equal to 1.

#### 6.2.1. Single drone scenario

When $m = 1$, the best performing algorithms are COL-S, BIN-S, and GLR-S in general, which substantially take into account the rewards in their delivery selection rule. Instead, GSW-S and GCRP-S, whose selection rule is unlinked to the rewards, show the worst performance. This trend is particularly emphasized for the Zipf parameter $\theta = 0.8$. In this case, there are a few large-value rewards and many small-value rewards. Thus, GLR-S is facilitated in its choice. When the rewards are uniformly distributed ($\theta = 0$), COL-S guarantees better performance than GLR-S, probably because the solution considers also the compatibility of the intervals, and not only blindly the reward. In fact, while COL-S first selects the optimal sets of compatible intervals and then chooses the best one (with the largest overall reward), GLR-S could blindly select only a few large intervals, among the best ones in terms of reward, compromising the possibility to extend the goodness of the solution even if the drone has more energy to spend. Instead, in presence of unbalanced rewards ($\theta = 0.8$), GLR-S exhibits the best solution because the algorithm can select the most profitable intervals without any other consideration. The performance jump of GLR-S is high when $\theta = 0.8$. The difference between the two approximation algorithms COL-S and BIN-S is large when the variability is low (e.g., $\Sigma_1$) with COL-S that obtains $\approx 85\%$ of the optimum,

while BIN-S gets $\approx$ 55% of the optimum reward. However, when the variability is higher (e.g., $\Sigma_4$), they almost perform the same (with BIN-S slightly better than COL-S). Another aspect to consider is the global loss of performance when $n$ increases. This is probably due to the increase of conflicts among the intervals, and therefore sub-optimal algorithms can suffer when selecting the deliveries for the drone without any conflict. Obviously, the ratio of KNA-S is always 1 since it is optimal.

*6.2.2. Multiple drones scenario*

In the multiple drone scenario, i.e., for $m = \{3, 5\}$, we observe that GCRP-M and GSW-M perform poorly, as in the single drone scenario. Accordingly, strategies that pick intervals (deliveries) taking into account either weight or rendezvous time do not perform well. The approximation algorithm KNA-M has an extremely good performance collecting $\geq$ 95% in any configuration. Although the guaranteed approximation ratio of KNA-M is $\frac{1}{m}$, where $m$ is the number of drones, its performance is much better than that threshold. This suggests to us that the approximation analysis for KNA-M can be probably improved. Experimentally, we have observed that KNA-M often returns the optimum, while in other cases there is only a little loss of performance, like $\approx$ 3% in average. Since KNA-M sequentially performs an optimal strategy on the residual subset of deliveries, almost always the reward obtained by the first drone is much larger than that of the second drone, and so on, while the reward that all the drones get in the optimal solution is more or less equally distributed among them (it is indeed pretty balanced). This last observation can help us find the optimal solution for the multiple drones case. Differently from the single drone case when BIN-S was employed, BIN-M performs much better in the multiple drones case when the variability is low (e.g., $\Sigma_1$). We get this behavior because most likely intervals in conflict can still be assigned to different drones, and the effect of having many intersections is less important. On the other hand, the COL-M heuristic algorithm consolidates the good performance obtained in single drone case by COL-S. In particular, we observe a quite stable trend on the four configurations which degrades on average variability configurations, i.e., $\Sigma_2$ and $\Sigma_3$. This happens probably because the coloring procedure, which computes $\chi$ feasible set of intervals, does not consider the reward to solve conflicts, therefore it may wrongly select low reward intervals at expense of more valuable intervals. However, COL-M experimentally guarantees $\approx$ 80% of the optimum in any configuration. Finally, we observe that the performances of our proposed algorithms with $m = 5$ drones are slightly better than those with $m = 3$. This is probably because the residual intervals can be assigned to other independent drones without conflicts. Notice that, we did not consider experiments with more than 5 drones because it would have been impractical to get the optimal solution computed by the CPLEX solver.

## 7. Conclusion

This paper investigated the Scheduling Conflictual Deliveries Problem (SCDP) to study the cooperation between a truck and multiple drones in a last-mile package delivery scenario. After showing that SCDP is an *NP*-hard problem, we proposed an ILP formulation that is suitable for small instances in input. We also gave an optimal pseudo-polynomial time algorithm for the single drone case and also provided additional time-efficient approximation and heuristic algorithms for the single and multiple drones cases. Finally, we evaluated the performances of the proposed algorithms on synthetic datasets. In future work, it would be interesting to investigate multi-depot multi-truck scenarios or allow drones to perform multiple deliveries at the same time or recharge their battery on charging stations. We also plan to study a more realistic dynamic environment, dealing with network communications between the depot and ground–air vehicles, and addressing possible delays and new/canceled deliveries. To address these challenges, we plan to develop online strategies to reschedule deliveries on the fly.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

Data will be made available on request.

## References

[1] A. Restas, et al., Drone applications for supporting disaster management, World J. Eng. Technol. 3 (03) (2015) 316.
[2] T. Calamoneri, F. Corò, S. Mancini, A realistic model to support rescue operations after an earthquake via UAVs, IEEE Access (2022).
[3] U.R. Mogili, B. Deepak, Review on application of drone systems in precision agriculture, Procedia Comput. Sci. 133 (2018) 502–509.
[4] A. Kadethankar, N. Sinha, V. Hegde, A. Burman, Signature feature marking enhanced IRM framework for drone image analysis in precision agriculture, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2021, pp. 2385–2389.
[5] F.B. Sorbelli, F. Corò, S.K. Das, L. Palazzetti, C.M. Pinotti, Drone-based optimal and heuristic orienteering algorithms towards bug detection in orchards, in: 18th Int. Conf. on Distr. Computing in Sensor Systems, DCOSS, IEEE, 2022, pp. 1–8.
[6] F.B. Sorbelli, S.K. Das, C.M. Pinotti, G. Rigoni, A comprehensive investigation on range-free localization algorithms with mobile anchors at different altitudes, Pervasive Mob. Comput. 73 (2021) 101383.

[7] E. Frachtenberg, Practical drone delivery, Computer 52 (12) (2019) 53–57.
[8] A. Khanda, F. Coro, F.B. Sorbelli, C.M. Pinotti, S.K. Das, Efficient route selection for drone-based delivery under time-varying dynamics, in: 2021 IEEE 18th Intl. Conf. on Mobile Ad Hoc and Smart Systems, MASS, IEEE, 2021, pp. 437–445.
[9] F.B. Sorbelli, C.M. Pinotti, G. Rigoni, On the evaluation of a drone-based delivery system on a mixed euclidean-manhattan grid, IEEE Trans. Intell. Transp. Syst. (2022).
[10] A.M. Ham, Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming, Transp. Res. C 91 (2018) 1–14.
[11] D.N. Das, R. Sewani, J. Wang, M.K. Tiwari, Synchronized truck and drone routing in package delivery logistics, IEEE Trans. Intell. Transp. Syst. 22 (9) (2021) 5772–5782.
[12] K. Nonami, Drone technology, cutting-edge drone business, and future prospects, J. Robot. Mechatron. 28 (3) (2016) 262–272.
[13] R. Arnold, et al., Experimentation for optimization of heterogeneous drone swarm configurations: terrain and distribution, in: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III, Vol. 11746, International Society for Optics and Photonics, 2021, 1174625.
[14] J. Lee, Optimization of a modular drone delivery system, in: 2017 Annual IEEE Intl. Systems Conf., SysCon 2017, Montreal, QC, Canada, April 24-27, 2017, IEEE, 2017, pp. 1–8.
[15] I. Dayarian, M. Savelsbergh, J.-P. Clarke, Same-day delivery with drone resupply, Transp. Sci. 54 (1) (2020) 229–249.
[16] C.C. Murray, A.G. Chu, The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery, Transp. Res. C 54 (2015) 86–109.
[17] N. Mathew, S.L. Smith, S.L. Waslander, Planning paths for package delivery in heterogeneous multirobot teams, IEEE Trans. Autom. Sci. Eng. 12 (4) (2015) 1298–1308.
[18] S. Kim, I. Moon, Traveling salesman problem with a drone station, IEEE Trans. Syst. Man Cybern. 49 (1) (2018) 42–52.
[19] G.C. Crişan, E. Nechita, On a cooperative truck-and-drone delivery system, Proc. Comp. Sci. 159 (2019) 38–47.
[20] R. Daknama, E. Kraus, Vehicle routing with drones, 2017, CoRR arXiv:1705.06431.
[21] F. Betti Sorbelli, F. Corò, S.K. Das, L. Palazzetti, C.M. Pinotti, Greedy algorithms for scheduling package delivery with multiple drones, in: 23rd Intl. Conf. on Distributed Computing and Networking, 2022, pp. 31–39.
[22] R. Roberti, M. Ruthmair, Exact methods for the traveling salesman problem with drone, Transp. Sci. 55 (2) (2021) 315–335.
[23] N. Boysen, D. Briskorn, S. Fedtke, S. Schwerdfeger, Drone delivery from trucks: Drone scheduling for given truck routes, Networks 72 (4) (2018) 506–527.
[24] D. Wang, et al., Routing and scheduling for hybrid truck-drone collaborative parcel delivery with independent and truck-carried drones, Int. Things J. 6 (6) (2019) 10483–10495.
[25] Y.S. Chang, H.J. Lee, Optimal delivery routing with wider drone-delivery areas along a shorter truck-route, Expert Syst. Appl. 104 (2018) 307–317.
[26] C.C. Murray, R. Raj, The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones, Transp. Res. C 110 (2020) 368–398.
[27] S. Martello, D. Pisinger, P. Toth, New trends in exact algorithms for the 0–1 knapsack problem, European J. Oper. Res. 123 (2) (2000) 325–332.
[28] F. Gavril, Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, SIAM J. Comput. 1 (2) (1972) 180–187.
[29] T.M. Chan, Approximation schemes for 0-1 knapsack, in: 1st Symposium on Simplicity in Algorithms (SOSA 2018), Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018, p. 1.
[30] A. Caprara, H. Kellerer, U. Pferschy, D. Pisinger, Approximation algorithms for knapsack problems with cardinality constraints, European J. Oper. Res. 123 (2) (2000) 333–345.
[31] G.L. Nemhauser, L.A. Wolsey, M.L. Fisher, An analysis of approximations for maximizing submodular set functions-I, Math. Program. 14 (1) (1978) 265–294.
[32] M. Sviridenko, A note on maximizing a submodular set function subject to a knapsack constraint, Oper. Res. Lett. 32 (1) (2004) 41–43.
[33] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2009.
[34] J. Kleinberg, E. Tardos, Algorithm Design, Pearson Education India, 2006.
[35] J.M. Harris, J.L. Hirst, M.J. Mossinghoff, Combinatorics and Graph Theory, second ed., in: Undergraduate Texts in Mathematics, Springer, 2008.
[36] S.S. Seiden, On the online bin packing problem, J. ACM 49 (5) (2002) 640–671.
[37] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, R.L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Comput. 3 (4) (1974) 299–325.
[38] D.S. Johnson, Near-optimal bin packing algorithms (Ph.D. thesis), Massachusetts Institute of Technology, 1973.
[39] M. Qasim, C. Csaba, Major barriers in adoption of electric trucks in logistics system, Promet-Traffic Transp. 33 (6) (2021) 833–846.
[40] J.K. Stolaroff, C. Samaras, E.R. O'Neill, A. Lubers, A.S. Mitchell, D. Ceperley, Energy use and life cycle greenhouse gas emissions of drones for commercial package delivery, Nature Commun. 9 (1) (2018) 1–13.
[41] F.B. Sorbelli, F. Corò, S.K. Das, C.M. Pinotti, Energy-constrained delivery of goods with drones under varying wind conditions, IEEE Trans. Intell. Transp. Syst. 22 (9) (2021) 6048–6060.
[42] C. Tullo, J. Hurford, Modelling zipfian distributions in language, in: Proceedings of Language Evolution and Computation Workshop/Course at ESSLLI, 2003, pp. 62–75.