Wireless IoT sensors data collection reward maximization by leveraging multiple energy- and storage-constrained UAVs

Wireless IoT Sensors Data Collection Reward Maximization by Leveraging Multiple Energy- and Storage-Constrained UAVs ⋆

Francesco Betti Sorbelli[a], Alfredo Navarra[a], Lorenzo Palazzetti[b,a], Cristina M. Pinotti[a], Giuseppe Prencipe[c]

[a] *Department of Computer Science and Mathematics, University of Perugia, Italy*
[b] *Department of Computer Science and Mathematics, University of Florence, Italy*
[c] *Department of Computer Science, University of Pisa, Italy*

Abstract

We consider Internet of Things (IoT) sensors deployed inside an area to be monitored. Drones can be used to collect the data from the sensors, but they are constrained in energy and storage. Therefore, all drones need to select a subset of sensors whose data are the most relevant to be acquired, modeled by assigning a reward. We present an optimization problem called *Multiple-drone Data-collection Maximization Problem* (MDMP) whose objective is to plan a set of drones' missions aimed at maximizing the overall reward from the collected data, and such that each individual drone's mission energy cost and total collected data are within the energy and storage limits, respectively. We optimally solve MDMP by proposing an Integer Linear Programming based algorithm. Since MDMP is *NP*-hard, we devise suboptimal algorithms for single- and multiple-drone scenarios. Finally, we thoroughly evaluate our algorithms on the basis of random generated synthetic data.

Keywords: Drones, Sensor networks, Data collection, Integer Linear Programming, Approximation algorithms

1. Introduction

With the recent advent of the Unmanned Aerial Vehicles (UAVs), such as drones, we have seen a rapid growth of civilian applications that make use of them. Just to mention a few, drones can be efficiently employed for localizing missing people in search and rescue operations [2, 3], for delivering goods in the last-mile scenario [4, 5, 6], for scouting insects in a precision agriculture context [7, 8]. In this last smart agriculture application, drones can plan missions inside an orchard to take pictures of trees at different heights (to photograph as many bugs as possible, if present) so that subsequent machine learning algorithms can recognize the abundance of the pest [9]. The

⋆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, by MIPAAF, by RESIDUAL, and by RB_DMI_2019. Part of this work has been accepted [1] to the 2022 International Symposium on Algorithmics of Wireless Networks (ALGOSENSORS 2022).

Email addresses: francesco.bettisorbelli@unipg.it (Francesco Betti Sorbelli), alfredo.navarra@unipg.it (Alfredo Navarra), lorenzo.palazzetti@unifi.it, lorenzo.palazzetti@collaboratori.unipg.it (Lorenzo Palazzetti), cristina.pinotti@unipg.it (Cristina M. Pinotti), giuseppe.prencipe@unipi.it (Giuseppe Prencipe)

main reason why drones are used in such types of applications is their ability to perform very challenging tasks in an easy way. In fact, autonomous ground vehicles, such as robots, can still execute tasks on their own, but they are particularly constrained by mobility. For instance, it is really hard for a robot, if not impossible, to overtake a creek or travel inside a plough land, whereas for drones, this task can be quickly done. Even inside an orchard (or vineyard), a robot has to go either at the beginning or at the end of a row to swap the monitored row [10], while for a drone, this task would be significantly easier because it can overfly the row.

With the current state of technology, multiple applications can be done in smart agriculture with the help of drones. In addition to the aforementioned bug detection application, also the spread of pesticides and insecticides on crops, the monitoring of the health of the field, and also the collection of sensed data from ground Internet of Things (IoT) sensors, can be done by leveraging drones. Taking into account the latter application, farmers are starting to rely on these IoT sensors to implement smart agriculture. In particular, IoT sensors [11] can simply sense the current temperature or air humidity, can determine the current wind speed and direction, or can even take pictures or record videos by using RGB or infrared chips. Since these sensors have a limited radio communication range [12] and available storage [13], detected data must be periodically transferred to an external device for further analysis [14]. Furthermore, the deployment area can be very large and therefore the sensors cannot directly transfer their perceived data to the main base station [15] (briefly, the depot). Even with a multi-hop paradigm, there is still the issue that sensors closer to the depot use more energy while relaying data [16], and the network can be easily disconnected.

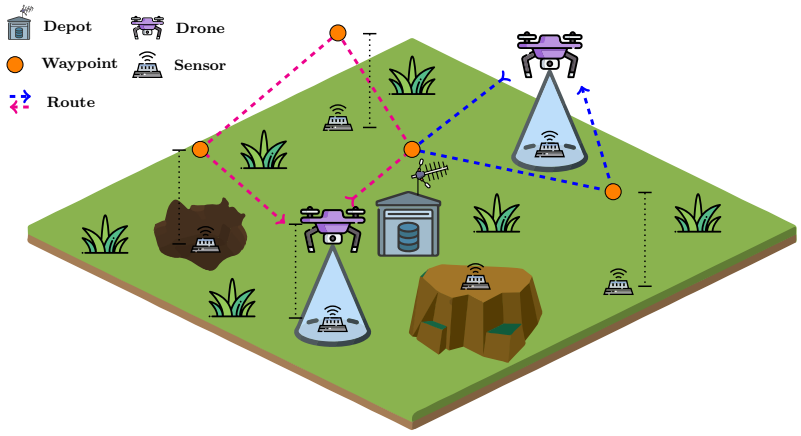

Figure 1: The sketched representation of our application. The surface is not flat, and therefore the sensors have different heights with respect to the depot.

In this work, we consider that a set of multiple homogeneous drones is responsible for collecting data from ground IoT sensors. An example of a data collection scenario using two drones is depicted in Figure 1. We assume that a set of heterogeneous ground sensors is randomly deployed on an area for sensing particular phenomena. Due to the fact that the deployment area can be very large, the sensors cannot directly transfer their perceived data to the depot. In our proposed architecture, drones flying over the area are responsible for performing a mission (a route) to/from the depot, with the objective of selectively collecting the data from the sensors. However, the drones themselves are limited in terms of energy battery (when flying and hovering) and available memory storage (when collecting data). In principle, due to the two limitations mentioned above,

drones cannot collect all the data from all deployed sensors, but they have to plan a proper route and use parsimoniously both their available energy and storage. Moreover, in harsh environments, the drones could not immediately transmit the collected data to the cloud/depot because the Internet connectivity can be absent, and hence they need to keep the data to their storage, which is limited in capacity, until they finally reach the main depot.

In our proposed context, certain sensors' data are more critical and should be collected with higher priority than others. This implies that some data are more relevant and important than others. Moreover, since some older data can be lost if not offloaded in a timely manner, it is crucial to consider data more relevant if stored in a sensor that has less space left. Therefore, the farmer of the field to be monitored must prioritize these sensors over others, based on the criticality of the data and the likelihood of data loss. To model the relevance and consequent prioritization of data to be collected by drones, a specific reward is assigned to each sensor data based on the data relevance. Urgent data requiring immediate analysis is given a higher reward than regular data, and data that may be lost due to the shortage of available local storage is also prioritized.

The primary goal pursued in this paper is maximizing the total reward obtained by collecting the most relevant data using a fleet of drones. However, this must be achieved while ensuring that the energy cost of each drone's mission does not exceed the battery budget, and that the total collected data do not exceed the storage limit on each drone. To the best of our knowledge, this is the first time that a fleet of homogeneous drones is in charge of collecting data from a set of heterogeneous ground sensors while simultaneously taking into account both the available energy and storage for the drones.

This work extends our previous conference paper [1] in which only a single drone was involved in data collection. The contributions of this paper are summarized below.

- We define a novel optimization problem, called *Multiple-drone Data-collection Maximization Problem* (MDMP), whose goal is to collect the most relevant data by leveraging drones, i.e., maximizing the total obtained reward, while ensuring that each drone's mission energy cost does not exceed the battery budget, and the total collected data do not exceed the storage limit on each drone;
- We formally prove that MDMP is *NP*-hard (even for the single-drone scenario) because the Team Orienteering Problem (TOP) can be seen as a particular instance of MDMP;
- We devise an Integer Linear Programming (ILP) formulation for optimally solving MDMP, only suitable for small-sized inputs;
- We propose approximation and heuristic algorithms for obtaining suboptimal solutions for the single- and multiple-drone scenarios for inputs of any size;
- We thoroughly evaluate the performance in terms of collected reward of our algorithms on randomly generated synthetic data.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 formally defines MDMP showing its *NP*-hardness, and proposes the ILP-based optimal algorithm for it. Section 4 and Section 5 present suboptimal approximation and heuristic algorithms for solving MDMP for the scenario of a single and multiple drones, respectively. Section 6 evaluates the effectiveness of our algorithms on randomly generated synthetic data, and Section 7 offers conclusions and future research directions.

2. Related Work

Many papers have been proposed in the realm of data collection in sensor networks with the help of drones.

In [17, 18], the authors consider the problem of scheduling the flight of a drone in charge to maximize the utility due to the data collected in a sensor network composed by homogeneous sensors deployed on a flat surface. The drone has the ability to simultaneously collect data from multiple sensors and its hovering time depends on the size of data to be collected. The authors discretize the possible hovering points for the drone in order to limit the number of them. A similar scenario is also considered in [19], in which the objective is to find the tour that maximizes the utility of the collected data considering the data transfer divided into time slots of equal width. Furthermore, the paper in [20] considers the problem of determining the minimum number of UAVs to be deployed to collect all the data from sensors on a flat area without exceeding a given budget time. Two algorithms are proposed to solve the problem. In contrast to the three previously mentioned works, our paper considers several crucial factors that have been not accounted for, namely the presence of heterogeneous sensors at different elevations, the energy consumption required for flight, the drone's storage capacity limitation, as well as the presence of a fleet of multiple drones. Furthermore, to prevent potential bandwidth saturation, we do not permit the simultaneous collection of data from multiple sensors. These considerations are of significant importance when designing a drone-based data collection system, and our work addresses these challenges in a comprehensive manner.

The problem of scheduling the UAV's tour which minimizes the maximum energy consumption for all the sensors is studied in [21]. The authors jointly consider the sensors' wake-up and the UAV's path by formulating a mixed-integer non-convex optimization program, and a suboptimal algorithm which iteratively applies a successive convex optimization technique. The authors in [22] propose a clustering algorithm and meta-heuristics to address a similar problem where the sensors are deployed in a hilly terrain, and the single UAV is not energy-constrained. In contrast to the above papers, which consider homogeneous sensors relying on continuous communications or even not energy-constrained drones, our paper proposes approximation and time-efficient heuristic solutions for a system with multiple drones and heterogeneous sensors. Additionally, all the previous works only offer computationally expensive exact solutions or meta-heuristic solutions, whereas we also propose more time-efficient deterministic methods.

The problem of maximizing the *freshness* of the data collected by a UAV has been studied in [23]. In particular, two problems of Age-of-Information (AoI) data collection are formulated to minimize both the sensors' maximal and average AoI. In [24], the authors propose a framework for controlling the flight speed of the UAV to improve the *fairness* of data collection. They formalize fairness as a metric that depends on the energy level of the sensor nodes and on the amount of data to be sent to the UAV. Specifically, since only cluster heads have to transfer data collected via multi-hop from the other nodes to the UAV, their fairness is the least. Therefore, the authors develop a method which controls and adjusts the UAV's speed according to the intra-cluster density of sensors, and the distance from the UAV and each sensor. However, unlike our work which deals with multiple drones each with energy and storage constraints, the authors do not consider the energy consumption and the storage availability of the single drone in their approach.

The authors in [25] present an optimization problem where a drone is in charge to collect data from a set of ground sensors. Here, the locations where the drone has to stop are not known in advance, and they are computed according to a clustering scheme. The goal is to determine

the single drone's path such that the drone's energy is minimized. When computing the path, a trade-off between the flight duration and the communication reliability is required. The problem is solved relying on ILP formulations. A similar approach is proposed in [26], in which the goal is only to minimize the flight time of the single drone. The problem is optimally solved by performing a brute-force algorithm plus two heuristic algorithms. The above papers differ from our approach in that they only utilize a single drone, and they do not take into account data relevance in their modeling. Consequently, their objective is to maximize the quantity of collected data, whereas our approach is a more generalized version in which are involved multiple drones, and incorporated data relevance as a factor in our optimization.

In [27], the data collection problem is investigated under a "security" point of view. In fact, a fleet of drones is responsible for selectively collect data from a set of ground sensors, with the main goal to guarantee the security of the stored data. Different metrics are optimized in [27], such as computational cost, energy consumption, and communication overhead. The optimization function used in their approach did not take into consideration any kind of relevance of the data, as well as the storage of the drones.

The authors of [28] investigate data collection in a wireless sensor network by using multiple drones. Two main issues are addressed: (1) how to ensure seamless synchronization among sensors and drones at each transit, and (2) how to compute energy-efficient schedules for the sensors and feasible trajectories for the drones. To solve the problem, a joint wake-up scheduling and drone path planning optimization problem is formulated; eventually, simulations are also proposed. Similarly, in [29] drones are used to gather data from a set of ground sensors that rely on a Long Range Wide Area Network (LoRaWAN) protocol. The proposed problem takes into account three objectives, such as: (1) minimize the total drone's flying time, (2) collect all data packets from all nodes, and (3) minimize the nodes' energy consumption. Differently from our approach, the authors focus on prolonging the lifetime of the sensor network by only considering the limited battery capacity of the drones, without accounting for the storage constraint or the relevance of the collected data.

Finally, a recent work in [30] proposes an optimization problem to find a sequence of locations that the drone should follow inside the monitored area, so that the overall time to collect data is minimized. Unlike us, the authors only use a single drone and collect data from all sensors, without considering the relevance of their data or selecting a subset of sensors based on their relevance.

3. Problem Definition

In this section, we introduce the system model, define our novel problem for data collection in an IoT sensor network by leveraging drones, prove its *NP*-hardness, and devise the optimal solution for it by formulating an ILP.

3.1. System Model

Let F be the *field*, whose center $O = (0, 0, 0)$ is the *depot*, to be monitored by a set $V = \{v_1, \ldots, v_n\}$ of n heterogeneous IoT *ground sensors*. Each sensor $v_i \in V$ is randomly deployed in F, and its position is (x_i, y_i, z_i) with respect to O. The field F can be seen as a *complete graph* $G = (V, E)$ where the set V is the set of vertices/sensors, and the set E represents the edges/connections among each pair of sensors. The sensors collect data like the temperature, pressure, or even pictures or videos to be saved on their local storage of size W_i, assumed to be different for each sensor. In fact, tiny sensors that record text data have a small storage, but camera sensors that have to store large videos, need a much larger storage. Since W_i is limited, sensors

have to periodically transfer the recorded data to external devices. Let $0 < w_i \leq W_i$ be the *size* of the data that each v_i needs to transfer. As mentioned above, the data are modeled by a *relevance*, and the relevant data should be prioritized when ground sensors have to start data transfer. This is modeled by associating a *reward* $r_i > 0$ to each sensor v_i. Data that need immediate analysis are prioritized with a higher reward than standard data, and data that may be lost (overwritten) due to limited storage capacity are also associated with high priority. So, the more is the reward, the more relevant is to off-load the data to an external device. Importantly, given an instance of the problem, the relevance does not change, and remains the same until the mission is completed.

The external devices that collect sensor data are a set of l homogeneous *drones* denoted as $D = \{d_1, \ldots, d_l\}$. The *flight mission* of each drone starts and finishes at O. The drones fly at a fixed altitude h above the ground and have a *communication range* with a radius R. So, a drone $d_k \in D$ can collect data from a sensor v_i if $\|d_k - v_i\|_2 \leq R$, i.e., if their relative *Euclidean distance* is within the communication range. In this work, we neglect communication issues such as shadowing, fading, or multipath propagation. Moreover, we assume that the drones and the sensors are always in line of sight and hence no obstacles are present. We also assume that the drones cannot collide among them or with any sensor. In fact, by opportunely tuning the heights at which drones fly, we can assume that drones cannot collide. However, for the sake of clarity, we simply consider the same height for all the drones even though collisions are still not addressed.

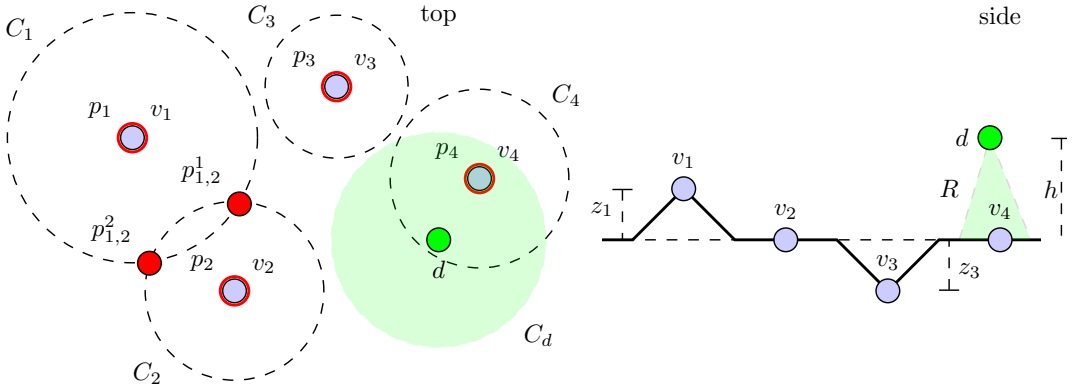

Figure 2: Top and side representations of the field F.

The drones are allowed to hover only at specific locations over F, called *waypoints*, represented by a set P of possible positions. Firstly, P contains all the positions of the projected sensors at height h, i.e., $\forall v_i = (x_i, y_i, z_i) \in V$ we have $p_i = (x_i, y_i, h) \in P$. P also contains other points as follows. For each sensor v_i, we define an admissible region in which the drones can actually communicate with it. Such a region is delimited by a circumference C_i of radius $\leq \sqrt{R^2 - (h - z_i)^2}$. To be more precise, a drone flying at a height h has a communication sphere of radius R which intersects the ground level ($h = 0$) forming a circumference C_d of radius $\leq \sqrt{R^2 - h^2}$. A sensor can communicate with a drone if the center of C_i is inside the circumference C_d. E.g., in Figure 2 the sensor v_4 and the drone d can communicate. In general, the number of possible drone waypoints can be unbounded. Therefore, in order to bound the number of waypoints for the drone [19], for each pair of sensors v_i and v_j, we add in P all the possible intersections $p^1_{i,j}$ and $p^2_{i,j}$ between C_i and C_j (see Figure 2). Also the depot $O = p_0 \in P$. So, given the n sensors, the number of waypoints is $m = |P| \leq n + n(n-1) + 1$, because two sensors can generate no more than two intersections.

Each individual drone is constrained by the limited energy of its battery that consumes when *moves* between locations and when *hovers* at waypoints. Sensors can start the data transfer procedure only when the drone hovers at waypoints. Hence, the sensors cannot transfer the data if a drone is currently moving even if their relative distance is within the communication range. We also assume that a drone cannot concurrently collect data from multiple sensors, but separately one at a time. So, if two sensors v_1 and v_2 are in range with a drone (say d_a), only one (say v_1) can transfer the data, while the other (say v_2) has to wait until v_1 finishes the procedure. However, if there is another drone (say d_b) in range with v_1 and v_2, and v_1 is transmitting towards d_a, d_b could collect the data from v_2 at the same time because there is not any conflict nor any queue among them. Moreover, we do not allow for a partial transferring, so when a sensor v_1 starts to transfer data to a drone d_a, d_a necessarily must hover at the waypoint until the data is completely collected. This also means that a sensor v_1 cannot transfer a portion of data to a drone d_a, and the remaining portion to another drone d_b.

Given a waypoint $p_i \in P$, let $\mathcal{Q}_i$ be the set of ground sensors possibly in range with any drone, i.e., *covered* by a drone if it flies in p_i. So, each sensor $v_k \in \mathcal{Q}_i$ can communicate with a drone in p_i because $\|p_i - v_k\|_2 \leq R$.

Let $\mathcal{E}^F(p_i, p_j)$ be the drone required *flying energy* for moving from p_i to p_j, which depends on the Euclidean distance between waypoints, i.e., $\|p_i - p_j\|_2$, and on the *energy per unit distance* parameter $\alpha > 0$, so that:

$$\mathcal{E}^F(p_i, p_j) = \alpha \cdot \|p_i - p_j\|_2. \tag{1}$$

Let $\mathcal{E}^H(p_i, t)$ be the required drone's *hovering energy* for statically staying at the waypoint p_i for t time slots, which depends on the number of time slots, i.e., t, and on the *energy per time slot* parameter $\beta > 0$, so that $\mathcal{E}^H(p_i, t) = \beta \cdot t$. However, for a given p_i, the number of the needed time slots t depends on the size of the data to be collected from the sensors in $\mathcal{Q}_i$. Recall that there are multiple drones, and a sensor can transfer its data to at most one drone. So, we have $\mathcal{Q}_i \supseteq \bigcup_{j=1}^{l} Q_i^j$, where Q_i^j is the subset of sensors that the drone d_j needs to collect the data from. Moreover, $\mathcal{Q}_i$ is a partition and hence we also have $Q_i^a \cap Q_i^b = \varnothing$ for any $a \neq b$. Thus, if there are sensors $v_k \in Q_i^j \subseteq \mathcal{Q}_i$, the cumulative data to transfer to the drone d_j from the sensors in Q_i^j at the waypoint p_i is $\sum_{v_k \in Q_i^j} w_k$. So, the required time depends on the total quantity of data to be transferred, and on the *data-transfer rate* parameter $\gamma_k > 0$ for the sensor v_k, so that we can finally redefine the hovering energy function as:

$$\mathcal{E}^H(Q_i^j) = \sum_{v_k \in Q_i^j} \frac{\beta}{\gamma_k} w_k. \tag{2}$$

Let $\mathcal{M} = \{M_1, \ldots, M_l\}$ be the set of the l drones' *missions*. Each individual mission M_j, accomplished by the drone d_j, is formed by a sequence of distinct waypoints to be visited to/from the depot $O = p_0$; i.e., M is a sequence $p_0, \ldots, p_i, \ldots, p_0$. For each waypoint p_i, the drone d_j actually obtains the data from a subset $Q_i^j \subseteq \mathcal{Q}_i$ of sensors due to its storage limitation. Finally,

$$\mathcal{C}_{M_j} = \mathcal{C}_{M_j}^F + \mathcal{C}_{M_j}^H \tag{3}$$

is the *total mission cost* (in terms of energy) of the mission M_j, where $\mathcal{C}_{M_j}^F$ is the *flying cost*, and $\mathcal{C}_{M_j}^H$ is the *hovering cost*. Therefore, given the sequence of consecutive waypoints $(p_i, p_k) \in M_j$, the

flying and hovering energy costs are, respectively,

$$\mathcal{C}^F_{M_j} = \sum_{(p_i,p_k)\in M_j} \mathcal{E}^F(p_i,p_k) \tag{4}$$

$$\mathcal{C}^H_{M_j} = \sum_{p_i\in M_j} \mathcal{E}^H(Q^j_i) \tag{5}$$

The other aspect to consider is the overall transferred data from the sensors to the drones. Namely, let $\mathcal{U}_{M_j}$ be the *total used storage* by the drone d_j when doing the mission M_j, i.e.,

$$\mathcal{U}_{M_j} = \sum_{p_i\in M_j, v_k\in Q^j_i} w_k. \tag{6}$$

Finally, let $\mathcal{R}_{M_j}$ be *total obtained reward* by the drone d_j when doing the mission M_j, i.e.,

$$\mathcal{R}_{M_j} = \sum_{p_i\in M_j, v_k\in Q^j_i} r_k. \tag{7}$$

Note that, when $l = 1$, Eqs. (3) (6) (7) can be rewritten as $\mathcal{C}_M$, $\mathcal{U}_M$, and $\mathcal{R}_M$, respectively.

Concerning the drones' constraints, let $\mathbb{E} > 0$ be the *available energy budget* on the battery for performing a mission, i.e., each drone has its own energy budget $\mathbb{E}$ because the drones are assumed to be homogeneous. Moreover, let $\mathbb{S} > 0$ be the *available storage budget* on the mass storage for collecting the sensors' data. Again, each drone has its own storage budget $\mathbb{S}$. So, for each drone d_j, it is jointly required that:

$$\mathcal{C}_{M_j} = \mathcal{C}^F_{M_j} + \mathcal{C}^H_{M_j} \leq \mathbb{E} \tag{8}$$

$$\mathcal{U}_{M_j} \leq \mathbb{S}. \tag{9}$$

In this paper, we assume that any mission formed by *only a single waypoint* is feasible for both energy and storage for a single drone. Specifically, all sensors in the vicinity of such a waypoint can safely offload their data to a drone.

3.2. The Multiple-drone Data-collection Maximization Problem

In this paper, we present the *Multiple-drone Data-collection Maximization Problem* (MDMP) whose goal is *to find a set of routes for the drones to/from the depot, and a selection of sensors to assign to each drone, such that the sum of the total collected reward is maximized, and both the energy and storage budgets on each drone are not exceeded.* Given the set V of n sensors, the set D of l drones, and the energy and storage budgets of the drone $\mathbb{E}$ and $\mathbb{S}$, respectively, the objective is to determine the optimal set of missions $\mathcal{M}^*$ such that:

$$\mathcal{M}^* = \arg\max \sum_{d_j\in D} \mathcal{R}_{M_j} \ : \ \mathcal{C}_{M_j} \leq \mathbb{E},\ \mathcal{U}_{M_j} \leq \mathbb{S}. \tag{10}$$

Now, we are in a position to show that:

Theorem 1. *The* MDMP *is NP-hard.*

Proof. The classical Team Orienteering Problem (TOP), which has been proven to be *NP*-hard [31], can be seen as a particular instance of MDMP. Recall that in TOP the goal is to find a set of suitable closed routes for a given fleet of vehicles inside a weighted graph such that the sum of the total collected reward on visited vertices is maximized, and the traveling route cost of each vehicle along edges is within the given budget in input. However, in TOP, there is no storage constraint. So, we can reduce any instance of TOP to MDMP as follows: The total available storage constraint for a single drone can be relaxed by setting $\mathbb{S} = +\infty$. Concerning the reward, in TOP there is no choice to perform in each waypoint. This corresponds to assume that $Q_i^j = \mathcal{Q}_i$ in MDMP, i.e., we select for the drone d_j all the reachable sensors at $p_i \in P$. After these modifications, any instance of TOP is exactly an instance of MDMP. So MDMP is *NP*-hard. When only one drone is considered, TOP becomes the Orienteering Problem (OP) which has been shown to be *NP*-Hard [31]. Hence, even for the single-drone scenario, MDMP is *NP*-Hard. □

In the next section, we will present the optimal algorithm capable of optimally solving MDMP.

3.3. ILP Formulation

The MDMP can be optimally solved using an ILP formulation. We enumerate the sensors as $\mathcal{V} = \{1, \ldots, n\}$, the waypoints as $\mathcal{P} = \{0, \ldots, m\}$ (0 is the depot), and the drones as $\mathcal{D} = \{1, \ldots, l\}$. Let $x_{ij}^k \in \{0, 1\}$ be a decision variable that is 1 if the sensor $i \in \mathcal{V}$ transfers its data to the drone $k \in \mathcal{D}$ at the waypoint $j \in \mathcal{P}$; otherwise, it is 0. Let $y_{\eta j}^k \in \{0, 1\}$ be a decision variable that is 1 if the drone $k \in \mathcal{D}$ travels from the waypoint $\eta \in \mathcal{P}$ to the waypoint $j \in \mathcal{P}$; otherwise it is 0. Let $1 \leq u_i^k \leq m$ be a dummy variable that indicates the temporal order of the waypoints visited by the drone $k \in \mathcal{D}$, i.e., $u_\eta^k < u_j^k$ waypoint η is visited by the drone k before the waypoint j [32]. So, the ILP formulation is:

$$\max \sum_{k=1}^{l} \sum_{i=1}^{n} \sum_{j=0}^{m} r_i x_{ij}^k \tag{11}$$

subject to:

$$\sum_{k=1}^{l} \sum_{j=0}^{m} x_{ij}^k \leq 1, \qquad \forall i \in \mathcal{V} \tag{12}$$

$$\sum_{j=1}^{m} y_{0j}^k = \sum_{\eta=1}^{m} y_{\eta 0}^k = |\mathcal{D}|, \qquad \forall \eta, j \in \mathcal{P} \setminus \{0\}, \forall k \in \mathcal{D} \tag{13}$$

$$y_{jj}^k = 0, \qquad \forall j \in \mathcal{P} \setminus \{0\}, \forall k \in \mathcal{D} \tag{14}$$

$$\sum_{\eta=1}^{m} y_{\eta\nu}^k = \sum_{j=1}^{m} y_{\nu j}^k = \max_{i \in \mathcal{V}} x_{i\nu}^k, \qquad \forall \nu \in \mathcal{P}, \forall k \in \mathcal{D} \tag{15}$$

$$u_\eta^k - u_j^k + 1 \leq m(1 - y_{\eta j}^k), \qquad \forall \eta, j \in \mathcal{P} \setminus \{0\}, \forall k \in \mathcal{D} \tag{16}$$

$$1 \leq u_\eta^k \leq m, \qquad \forall \eta \in \mathcal{P} \setminus \{0\}, k \in \mathcal{D} \tag{17}$$

$$\sum_{i=1}^{n} \sum_{j=0}^{m} w_i x_{ij}^k \leq \mathbb{S}, \qquad \forall k \in \mathcal{D} \tag{18}$$

Table 1: Table of Notation.

Symbol	Description
F	field to be monitored by the drone
$V = \{v_1, \ldots, v_n\}$	set of n ground sensors
(x_i, y_i, z_i)	position of sensor v_i with respect to the center of the field O
$G = (V, E)$	complete graph representing F
E	set of edges/connections among sensors in V
W_i	local storage of sensor v_i
$w_i \leq W_i$	size of data that sensor v_i needs to transfer
r_i	reward associated with sensor v_i (relevance)
$D = \{d_1, \ldots, d_l\}$	set of l drones
h	altitude of the drones above the ground with respect to O
R	communication range radius
P	set of possible drone waypoints
p_i	a waypoint representing the position of projected sensor v_i
C_i	communication circumference of sensor v_i
$p^1_{i,j}$, $p^2_{i,j}$	intersection points between C_i and C_j
m	number of waypoints in P
$\mathcal{Q}_i$	set of sensors possibly in range with a drone when flying in p_i
$\mathcal{E}^F(p_i, p_j)$	drone required flying energy for moving from p_i to p_j
α	energy per unit distance
$\mathcal{E}^H(p_i, t)$	required drone's hovering energy for staying at p_i for t time slots
t	number of time slots in which the drone hovers
β	energy per time slot
Q^j_i	subset of sensors that drone d_j in p_i needs to collect the data from
γ_k	data-transfer rate for sensor v_k
$\mathcal{E}^H(Q^j_i)$	redefined hovering energy function
$\mathcal{M} = \{M_1, \ldots, M_l\}$	set of the l drones' missions
$\mathcal{C}_{M_j}$	total mission cost (in terms of energy) of the mission M_j
$\mathcal{C}^F_{M_j}$	flying cost of mission M_j
$\mathcal{C}^H_{M_j}$	hovering cost of mission M_j
$\mathcal{U}_{M_j}$	total used storage by drone d_j when doing the mission M_j
$\mathcal{R}_{M_j}$	total obtained reward by drone d_j when doing the mission M_j
$\mathbb{E} > 0$	available energy budget on the battery for performing a mission
$\mathbb{S} > 0$	available storage budget on the mass storage for collecting the sensors' data
$\mathcal{M}^*$	optimal set of missions

$$\sum_{j=0}^{m}\left(\sum_{i=1}^{n} h_i x_{ij}^k + \sum_{\eta=0}^{m} f_{\eta j} y_{\eta j}^k\right) \leq \mathbb{E}, \qquad \forall k \in \mathcal{D} \tag{19}$$

The objective function is represented by Eq. (11) which maximizes the overall reward. About the constraints, Eq. (12) states that each sensor can transfer its data no more than one time; Eq. (13) forces that the each drone's route begins and ends at the depot; Eq. (14) forbids self loops; Eq. (15) guarantees that each generated path is a simple cycle which contains the selected sensors; Eq. (16) ensures that no more than a single loop is allowed for each drone [32]; Eq. (17) indicates the temporal order of the visited waypoints, i.e., $u_\eta^k < u_j^k$ if p_η is visited before p_j by the k^{th} drone [32]; Eq. (18) guarantees the storage constraint of each drone; Eq. (19) guarantees the energy constraint of each drone, where $h_i = \mathcal{E}^H(p_i, w_i) \geq 0$ is the drone's hovering cost for transferring the data from sensor i, and $f_{lj} = \mathcal{E}^F(p_l, p_j) \geq 0$ is the drone's flying cost for moving from waypoints p_l to p_j.

We denote this formulation by OPT. Since OPT is only suitable for small inputs, in the following, we propose faster suboptimal algorithms suitable for any input. Specifically, in Section 4 we first devise algorithms for the particular case of a single drone, while in Section 5 we propose algorithms for the general case of multiple drones.

Table 1 summarizes the notation that has been adopted in this paper.

4. Solving MDMP with a Single Drone

In this section, we propose an approximation algorithm, called *Reward-Storage-first Energy-then Optimization* (RSEO-S), and two greedy heuristic algorithms, called *Max ratio Reward-Energy* (MRE-S), and *Max ratio Reward-Storage* (MRS-S), respectively, in order to solve MDMP with a single drone, i.e., $l = 1$. Note that the suffix "-S" stands for "single-drone". Moreover, in this section we denote M as the single-drone mission, while in the next section we will denote $\mathcal{M}$ as the set of missions of the fleet of drones.

4.1. The RSEO-S *Algorithm*

In this section, we devise an approximation algorithm that suboptimally solves MDMP with a single drone, called *Reward-Storage-first Energy-then Optimization* (RSEO-S). It is split into two phases. In the first phase, we select a subset of sensors such that the collected reward is maximized while ensuring that the storage requirement is met. Once the selection of sensors is done, we choose the minimum number of waypoints capable of covering all the selected sensors. At these waypoints, we compute the minimum energy-cost traveling route to/from the depot. Notice that the resulting route might be energy-unfeasible. If it is so, in the second phase we reduce the main route into a smaller one by removing the waypoint that reduces the least the lost reward. So, after the removal of the waypoint (along with two edges), we need to add an edge so that the path remains closed. This strategy is repeatedly done until we reach an energy-feasible route. The pseudocode of RSEO-S is given in Algorithm 1.

The objective is to maximize the reward by considering the drone's storage, and initially neglecting the drone's energy. This is realized by approximating three classical *NP*-hard subproblems, namely the knapsack, the min-set cover, and the traveling salesman.

The RSEO-S algorithm works as follows. Let us now focus on the first phase. We initially determine a subset of sensors $V' \subseteq V$ such that the obtained reward is maximized and the storage

Algorithm 1: The RSEO-S Algorithm

1 $V' \leftarrow$ *knapsack*$(V, \mathbb{S})$
2 $P' \leftarrow$ *min-set-cover*(V', P)
3 $M \leftarrow$ *traveling-salesman*(P')
4 **while** $\mathcal{C}_M > \mathbb{E}$ **do**
5 $\quad p \leftarrow \arg\min_{p_i \in M} \mathcal{R}_M - \mathcal{R}_{M \setminus \{p_i\}}$
6 $\quad M \leftarrow M \setminus \{p\}$
7 **return** M

constraint $\mathbb{S}$ is satisfied by invoking the *knapsack* procedure [33] (Line 1). In the knapsack procedure, we are given a collection of objects (sensors), each one associated with a size (data) and a reward (relevance), and we are asked to select a subset such that the total reward is maximized, while the total size occupied does not exceed that of the knapsack (drone's storage capacity). Let V' be the set of selected sensors and P be the family of subsets derived from the waypoints. Then, recalling that a sensor can be reached from multiple waypoints, we minimize the number of waypoints to visit to cover the entire set V' by invoking the *min-set-cover* procedure [33], determining so a subset of $P' \subseteq P$ of waypoints with cardinality $|P'| \leq |V'|$ (Line 2). In the *min-set-cover* procedure, we are given a set V' (sensors) and a family of subsets on P (waypoints), and the requirement is to select the minimum number of subsets whose union equals V'. Finally, since the drone has to perform a mission M to/from the depot visiting the waypoints P', we try to minimize the energy required by performing the *traveling-salesman* procedure [33] (Line 3). In the traveling-salesman procedure, we are given the set of points $|P'|$ (waypoints) in the Euclidean plane and a starting position p_0, with the requirement to traverse a tour starting and ending in p_0 so that all points are reached once and the traveled distance (energy) is minimized.

If the mission M is energy-feasible, then M is returned, otherwise we have to reduce M by removing a vertex (along with two edges) during the second phase. From all the waypoints that form M, we remove the one that minimizes the loss of reward associated to the that waypoint (Line 5). When we remove such a waypoint, we need to add an edge that ensures the existence of a closed path to/from the depot. This is repeated until M is energy-feasible. Eventually, the solution M is returned (Line 7).

Theorem 2. RSEO-S *solves* MDMP *with a single drone with an approximation ratio of* $\frac{\psi}{\mu\phi}$ *where* $\mu\phi$ *is the number of waypoints returned by a* ϕ*-approximation algorithm for the min-set-cover whose optimal solution has* μ *elements, which cover the sensors selected by a* ψ*-approximation algorithm for the knapsack.*

Proof. The solution M depends on the computed cycle starting from the *knapsack* invocation (that returns a subset $V' \subseteq V$ of sensors), which in turn depends on the ψ-approximation algorithm for solving it [33]. Recall that, $\psi \leq 1$ because the *knapsack* is a maximization problem. Then, in order to reduce the number of waypoints from which we collect the data, we rely on a ϕ-approximation version of the *min-set-cover* that returns at maximum $|P'| \leq \mu\phi \leq |V'| \leq n$ waypoints [33], where μ is the minimum number of waypoints able to cover V'. Recall that $\phi \geq 1$ because the *min-set-cover* is a minimization problem. If the resulting cycle given by the *traveling-salesman* [33] is energy-feasible, the approximation ratio of RSEO-S would be directly ψ, otherwise we need to prune some vertices reducing so the goodness of the solution. So, assuming $\mathcal{R}(SOL)$, $\mathcal{R}(OPT_{\text{KP}})$, and $\mathcal{R}(OPT)$ as the reward collected by the solution M, by the *knapsack*, and by the optimum

algorithm, respectively, we can now prove that:

$$\mathcal{R}(SOL) \geq \frac{\psi}{\mu\phi}\mathcal{R}(OPT_{\text{KP}}) \geq \frac{\psi}{\phi\mu}\mathcal{R}(OPT).$$

The first inequality holds since we rely on a ψ-approximated solution provided by *knapsack*. Moreover, according to our assumption, the drone actually selects at least one waypoint. Therefore, by selecting the best waypoint among the $\mu\phi$ ones, the collected reward of $\mathcal{R}(SOL)$ is at least a fraction $\frac{1}{\mu\phi}$ of the ψ-approximated solution of the *knapsack*. Finally, since $\mathcal{R}(OPT) \leq \mathcal{R}(OPT_{\text{KP}})$ is clearly true, the last inequality is also satisfied. □

In the next, we discuss the time complexity. In this paper, we rely on the greedy strategy for *fractional knapsack* which requires $\mathcal{O}(n \log n)$ time and also guarantees a $\frac{1}{2}$-approximation [33], i.e., $\psi = \frac{1}{2}$. Recall that $|P'| \leq |V'| \leq n$. To implement *min-set-cover* we rely on a greedy strategy which takes $\mathcal{O}(m|V'|)$ (because m is the cardinality of the subsets of sensors given by the waypoints) and guarantees a $(\log |V'|)$-approximation [33], i.e., $\phi = \log |V'|$. Regarding *traveling-salesman*, we exploit Christofides' $\frac{3}{2}$-approximation algorithm [33] (although it does not affect our ratio), which takes $\mathcal{O}(|P'|^3)$. Finally, since M comprises of $\mathcal{O}(|P'|)$ edges, and considering that at each iteration we remove one vertex, the time required by the loop (Line 4) is $\mathcal{O}(|P'| \log |P'|)$. Thus, the overall time complexity of RSEO-S is $\mathcal{O}(n \log n + m|V'| + |P'|^3 + |P'| \log |P'|) = \mathcal{O}(n^3)$, and our approximation bound is bounded from below by $\Omega\left(\frac{1}{2\mu \log |V'|}\right)$.

4.2. The MRE-S *Algorithm*

In this section, we devise a heuristic algorithm that suboptimally solves MDMP with a single drone, called *Max ratio Reward-Energy* (MRE-S). MRE-S greedily adds, to the current solution the sensor whose ratio between the reward and the additional energy cost with respect to the current drone mission is the largest. When computing this ratio for any new sensor, we also have to consider the energy for going back to the depot, since the drone cannot remain without energy. Moreover, a sensor can be selected only if the current drone's residual storage is enough. The pseudocode of MRE-S is given in Algorithm 2.

Algorithm 2: The MRE-S Algorithm

```
1 M ← ∅, P̂ ← {p_0, p_1, ..., p_n} ⊆ P
2 while P̂ ≠ ∅ do
3     p ← arg max_{p_i ∈ P̂} r_i / (C_{M∪{p_i}} − C_M)
4     if C_{M∪{p}} ≤ 𝔼 and U_{M∪{p}} ≤ 𝕊 then
5         M ← M ∪ {p}
6     P̂ ← P̂ \ {p}
7 return M
```

Initially, the solution M is empty, and a subset of waypoints $\hat{P}$ perpendicular to the sensors is created (Algorithm 2, Line 1). Then, the main cycle starts (Line 2) evaluating all possible waypoints $\hat{P}$. Among them, we select the waypoint p_i whose sensor v_i has the largest ratio between the reward and the additional energy cost with respect to the current drone's mission (Line 3). This greedy selection is justified by the fact that we aim at maximizing the reward while trying to keep low the

energy consumption. Then we evaluate whether p can be added to the current solution M without violating both the energy and storage constraints (Line 4). In any case, p will not be considered anymore and removed from $\hat{P}$ (Line 6). Finally, the solution M is returned (Line 7).

Regarding the time complexity of MRE-S, since the number of waypoints is $n + 1$ (because we only considered waypoints perpendicular to the sensors), the main loop is repeated $\mathcal{O}(n)$ times (Line 2). Since the selection of the best waypoint (Line 3) takes into account at most $\mathcal{O}(n)$ waypoints in each iteration, the total time complexity of the MRE-S algorithm is $\mathcal{O}(n^2)$.

4.3. The MRS-S *Algorithm*

In this section, we devise a heuristic algorithm that suboptimally solves MDMP with a single drone, called *Max ratio Reward-Storage* (MRS-S). MRS-S, similar to MRE-S, is based on the highest reward-to-storage ratio (instead of reward-to-energy). The Line 3 in Algorithm 2 is replaced by $p \leftarrow \arg\max_{p_i \in \hat{P}} \frac{r_i}{w_i}$. Once the selection is done, MRS-S tries to add p to the solution by evaluating if the energy and storage constraints are satisfied or not. In either cases, p will not be considered anymore. Eventually, the solution is returned.

Unlike MRE-S, all ratios can be initially calculated once, and hence we can sort them in a decreasing manner, which takes $\mathcal{O}(n \log n)$. So, at each iteration of the algorithm, we extract the current best one in constant time. Therefore, its time complexity is $\mathcal{O}(n \log n + n) = \mathcal{O}(n \log n)$.

5. Solving MDMP with Multiple Drones

In this section, we extend the previous three algorithms for the single-drone case (in Section 4) to multiple-drones to solve MDMP in the general case, specifically the heuristic algorithms RSEO-M, MRE-M, and MRS-M, respectively. Moreover, we present two heuristic algorithms called *Span And Split* (SAS-M), and *Clusterize And Assign* (CAA-M). Note that the suffix "-M" stands for "multiple-drone".

5.1. The RSEO-M *Algorithm*

In this section, we devise a heuristic algorithm that suboptimally solves MDMP with multiple drones, called RSEO-M. It extends the RSEO-S algorithm to a multiple-drone scenario, so they work similarly. In fact, the idea is to initially select and assign a subset of sensors to each drone such that the sum of the total collected reward is maximized while ensuring that the storage requirement is met (neglecting so the energy). Furthermore, as for RSEO-S, for each drone, we reduce the number of required waypoints to cover its assigned subset of sensors, then we compute a tour that connects all the waypoints and eventually shrink that tour if it is energy-unfeasible. The pseudocode of RSEO-M is given in Algorithm 3.

The RSEO-M algorithm works as follows. Initially, the solution is empty (Algorithm 3, Line 1). Then we determine a collection of subsets of sensors $\mathcal{V}' = \{V'_1, \ldots, V'_l\}$ such that the sum of the reward obtained from each V'_j is maximized and the storage constraint $\mathbb{S}$ is satisfied, by invoking the *multi-knapsack* procedure [34] (Line 2). The multi-knapsack procedure is a generalization of the knapsack procedure extended to multiple knapsacks. After that, for each knapsack V'_j (Line 3) assigned to the drone d_j, we reduce the number of required waypoints (Line 4), we connect all of them (Line 5), and possibly shrink the tour created if it exceeds the energy budget (Line 6). Eventually, the solution is returned (Line 10).

In the next, we discuss the time complexity. In the RSEO-M algorithm we need to invoke the *multi-knapsack* procedure at the beginning, which is an *NP*-hard problem [34]. In particular,

Algorithm 3: The RSEO-M Algorithm

1 $\mathcal{M} \leftarrow \varnothing$
2 $\mathcal{V}' = \{V'_1, \ldots, V'_l\} \leftarrow$ *multi-knapsack*$(V, \mathbb{S}, l)$
3 **foreach** $V'_j \in \mathcal{V}'$ **do**
4 $\quad P'_j \leftarrow$ *min-set-cover*(V'_j, P)
5 $\quad M_j \leftarrow$ *traveling-salesman*(P'_j)
6 $\quad$ **while** $\mathcal{C}_{M_j} > \mathbb{E}$ **do**
7 $\quad\quad p \leftarrow \arg\min_{p_i \in M_j} \mathcal{R}_{M_j} - \mathcal{R}_{M_j \setminus \{p_i\}}$
8 $\quad\quad M_j \leftarrow M_j \setminus \{p\}$
9 $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \{M_j\}$
10 **return** $\mathcal{M}$

Chekuri et al. proposed a $(1 - \epsilon)$ polynomial-time approximation scheme (PTAS) which takes $n^{\mathcal{O}(1/\epsilon^8 \log(1/\epsilon))}$ time [35, 36]. However, in this paper we decided to rely on the fast heuristic algorithm proposed by Martello et al. whose time complexity is $\mathcal{O}(ln^2)$ [37, 38]. The other subprocedures, i.e., *min-set-cover* and *traveling-salesman*, are executed l times, but on inputs smaller than those in the RSEO-S algorithm, and in the worst case, the loop in (Line 3) takes $\mathcal{O}(ln^3)$ time. In conclusion, the overall time complexity of the RSEO-M algorithm is $\mathcal{O}(ln^3)$.

5.2. The MRE-M *Algorithm*

In this section, we devise a heuristic algorithm that suboptimally solves MDMP with multiple drones, called MRE-M. It extends the MRE-S algorithm to a multiple-drone scenario, and therefore they work similarly. In fact, MRE-M greedily adds to the current drone's solution among the available l ones, the waypoint whose ratio between the overall obtainable reward and the additional energy cost with respect to the current drone's mission is the largest. Basically, MRE-M sequentially invokes MRE-S for each drone at the residual waypoints. The pseudocode of MRE-M is given in Algorithm 4.

Algorithm 4: The MRE-M Algorithm

1 $\mathcal{M} \leftarrow \varnothing, \hat{P} \leftarrow \{p_1, \ldots, p_n\}$
2 **foreach** $d_j \in D$ **do**
3 $\quad M_j \leftarrow$ MRE-S$(\hat{P} \cup \{p_0\})$
4 $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \{M_j\}, \hat{P} \leftarrow \hat{P} \setminus M_j$
5 **return** $\mathcal{M}$

The MRE-M algorithm works as follows. Initially, the solution is empty (Algorithm 4, Line 1), and the set of current waypoints is also created. Then, for each drone d_j (Line 2), we iteratively invoke the MRE-S algorithm on the current set of waypoints $\hat{P}$ (Line 3). Specifically, we sequentially return an energy- and storage-feasible drone's mission M_j for each drone to/from the depot, and then we add it to the set of missions $\mathcal{M}$, as well as we update the remaining waypoints (Line 4). Eventually, the solution is returned (Line 5).

The MRE-S algorithm takes $\mathcal{O}(n^2)$ time. Therefore, since the MRE-M algorithm sequentially invokes MRE-M l times, in the worst case, the cost of MRE-M is $\mathcal{O}(ln^2)$.

5.3. The MRS-M *Algorithm*

In this section, we devise a heuristic algorithm that suboptimally solves MDMP with multiple drones, called MRS-M. It extends the MRS-S algorithm to a multiple-drone scenario, and therefore they work similarly. It exactly works as the MRE-M algorithm, with one exception, i.e., the Line 3 in Algorithm 4 is replaced by $M_j \leftarrow$ MRS-S$(\hat{P})$.

Since MRS-S is repeated l times, its time complexity is $\mathcal{O}(ln \log n)$.

5.4. The SAS-M *Algorithm*

In this section, we devise a heuristic algorithm that suboptimally solves MDMP with multiple drones, called *Span And Split* (SAS-M). The idea is to initially create a path that connects all the vertices (sensors) plus the depot. This path is nothing but the sequence of vertices obtained by visiting the minimum spanning tree, rooted at the depot itself. Such a path exists because we assume that any two vertices can be connected following the straight line that connects them in the Euclidean plane. Then, we split the obtained path into a set of energy- and storage-feasible tours to be assigned to the drones. The pseudocode of SAS-M is given in Algorithm 5.

Algorithm 5: The SAS-M Algorithm

```
1 M ← ∅
2 T ← min-spanning-tree(G)
3 N ← tree-visit(T, p_0)
4 while N ≠ ∅ do
5 |   L ← create-tour(N, p_0)
6 |   M ← M ∪ {L}, N ← N \ L
7 M ← best-l-missions(M)
8 return M
```

The SAS-M algorithm works as follows. Initially, the solution is empty (Algorithm 5, Line 1). Then, we determine the minimum cost tree T in terms of drone's traveling energy that connects all the sensors V plus the depot p_0, by invoking the *min-spanning-tree* procedure (Line 2). In the *min-spanning-tree* procedure, since we are given a weighted graph G with weights on edges that represent the energy cost of flying between any two sensors, the objective is to select the tree that spans all vertices while minimizing the total flying/traveling energy cost on the edges. Once the tree is built, we perform a *tree-visit* procedure, and a sequence of vertices N is generated in output (Line 3). After that, we begin splitting the sequence of vertices N in tours to/from the depot p_0 (Line 4). Specifically, through the *create-tour* procedure (Line 5), starting every time at the point p_0, we try to create a tour L by selecting the next available vertex $v \in N$ not already taken in other tours. A vertex v can be added to the current tour L if the travel cost to visit v and go back to p_0, plus the hovering cost at v, is within the energy budget (as well as the whole collected data is within the storage budget); otherwise, the current tour L is closed (Line 6). When $N = \varnothing$, we assign the best l tours in terms of the collected reward to the l drones (Line 7), and eventually we return the solution (Line 8).

In the next, we discuss about the time complexity. The *min-spanning-tree* procedure that we have used in this paper is Kruskal's implementation and requires $\mathcal{O}(|E| \log n)$ time [39], where $|E|$ is the number of edges of the (complete) graph. The *tree-visit* procedure visits the minimum spanning tree T. In this paper, we implemented the Depth-first search (DFS) and the Breadth-first

search (BFS): both take time $\mathcal{O}(n)$ on trees [40]. Now, the cycle in Line 4 sequentially considers the vertices given by the *tree-visit* procedure, one at the time. The vertices are n. Whenever a vertex is not anymore neither energy- nor storage-feasible, the current tour is closed (*create-tour* procedure). Hence, the cycle in Line 4 costs $\mathcal{O}(n)$. Finally, the selection of the best l submissions costs $\mathcal{O}(n \log n)$ due to the sorting procedure. In conclusion, the SAS-M algorithm costs $\mathcal{O}(|E| \log n + n + n \log n) = \mathcal{O}(n^2 \log n)$ since $|E| = \mathcal{O}(n^2)$.

5.5. The CAA-M *Algorithm*

In this section, we devise a heuristic algorithm that suboptimally solves MDMP with multiple drones, called *Clusterize And Assign* (CAA-M). The idea is to initially split the set of vertices into l partitions (one for each drone) trying to balance the total energy cost. Then, we make each partition energy- and storage-feasible by invoking the previous RSEO-S algorithm. The pseudocode of CAA-M is given in Algorithm 6.

Algorithm 6: The CAA-M Algorithm

```
1 M ← ∅
2 V' = {V'_1, ..., V'_l} ← clusterize(V, l)
3 foreach V'_j ∈ V' do
4 |   M_j ← RSEO-S(V'_j, 𝔼, 𝕊)
5 |   M ← M ∪ {M_j}
6 return M
```

The CAA-M algorithm works as follows. Initially, the solution is empty (Algorithm 6, Line 1). Then we split the set of sensors V into l partitions by performing the *clusterize* procedure (Line 2). The clusterization phase takes into account the energy cost, so the hope is to determine clusters that are immediately energy-feasible regardless of the storage constraint. In other words, each cluster should be considered a neighborhood. This is in contrast to the RSEO-M algorithm, in which the "clusterization" given by the *multi-knapsack* procedure guarantees to have storage-feasible tours, while neglecting their energy cost. Therefore, each cluster V'_j should be adequately reduced in terms of storage, and shrunk in terms of energy cost, through the RSEO-S algorithm (Line 4). Finally, the solution is returned (Line 6).

In the next, we discuss about the time complexity. The *clusterize* procedure is an implementation of k-means clustering, which aims to minimize the sum of the squares of the distances between each point and its closest center by locating k cluster centers. This can be achieved through various methods, including Lloyd's local search algorithm, MacQueen's algorithm, and Hartigan-Wong's algorithm [41]. In this paper, for its implementation easiness, we utilized Lloyd's implementation through the *clusterize* procedure, which has a time complexity of $\mathcal{O}(nl)$ [42].

Then, recalling that the RSEO-S algorithm costs $\mathcal{O}(n^3)$, the cost of running it l times is $\mathcal{O}(ln^3)$. So, the final time complexity of the CAA-M algorithm is $\mathcal{O}(ln^3)$.

6. Performance Evaluation

In this section, we evaluate the performance, in terms of the reward obtained, of the algorithms presented to solve MDMP. We implemented[1] our algorithms in Python language version 3.9, and

[1]The code is available on GitHub here: `https://github.com/TheAnswer96/JCSS`

run all the instances on an Intel i7-10genK computer with 16 GB of RAM. However, the ILP-based optimal OPT algorithm is implemented using IBM's ILOG CPLEX Optimizer solver v22.1 with Python used to wrap the objective and constraints and invoke the parallel solver.

In Section 6.2 we present the results with a single drone scenario, while in Section 6.3 we present the results with a multiple drone scenario. For the previous scenario, we compare the RSEO-S, MRE-S, and MRS-S algorithms with respect to OPT (the optimal one), while in the latter one, we compare the RSEO-M, MRE-M, MRS-M, SAS-M, and CAA-M algorithms with respect to OPT.

In Table 2, we compare the algorithms presented evaluating their time complexities.

Table 2: Comparison between the algorithms for solving MDMP.

	Algorithm	Section	Time Complexity
Single-drone	RSEO-S	4.1	$\mathcal{O}(n^3)$
	MRE-S	4.2	$\mathcal{O}(n^2)$
	MRS-S	4.3	$\mathcal{O}(n \log n)$
Multiple-drone	RSEO-M	5.1	$\mathcal{O}(n^3)$
	MRE-M	5.2	$\mathcal{O}(ln^2)$
	MRS-M	5.3	$\mathcal{O}(ln \log n)$
	SAS-M	5.4	$\mathcal{O}(n^2 \log n)$
	CAA-M	5.5	$\mathcal{O}(ln^3)$

6.1. Settings

The field F is a square of the side 5 km, where the depot is located at the center of it. We uniformly generate $n = \{10, \ldots, 200\}$ sensors whose height is $-5\,\text{m} \leq z_i \leq 5\,\text{m}$. We also have a fleet of $l = \{1, 2, 3, 4\}$ drones that will be used to collect data. Each sensor has $100\,\text{MB} \leq w_i \leq 1\,\text{GB}$ data to transfer. This is a reasonable assumption because text data and picture data have different sizes in general. Moreover, each sensor has an associated reward $1 \leq r_i \leq 10$ that models the relevance of the data, so $r_i = 1$ models the lowest priority, while $r_i = 10$ models the highest priority. Both w_i and r_i are generated according to the Uniform distribution.

The drones fly at a fixed altitude $h = \{10, \ldots, 45\}$ m and have a fixed communication range of $R = 50\,\text{m}$ [43]. Their storage is $\mathbb{S} = \{2, \ldots, 16\}\,\text{GB}$ and their battery capacity is $\mathbb{E} = \{2.5, 5, 10\}\,\text{MJ}$ [44]. We fix an average energy consumption for flying $\alpha = 200\,\text{J/m}$ and hovering $\beta = 700\,\text{J/s}$ [45]. The data transfer rate between drones and sensors is set to $\gamma_i = 9\,\text{MB/s}$ (Wi-Fi 4 standard) regardless of the actual distance.

In the next plots, each algorithm is tested with different configurations of parameters, and we plot the average of the results on 33 random instances along with their 95% confidence interval. The optimal OPT is only run with small instances in input. In particular, when we run the OPT algorithm, we report in the y-axis the ratio $\rho = \frac{\mathcal{R}(SOL)}{\mathcal{R}(OPT)}$, i.e., the ratio among the total reward collected by the compared algorithm, and the total reward collected by the optimal algorithm. Clearly, $0 \leq \rho \leq 1$ because MDMP is a maximization problem. However, for larger instances in input, the OPT algorithm starts to be unsuitable due to the large number of variables/constraints, and therefore we only compare the other proposed approximation and heuristic algorithms. Obviously, since we do not compute the optimal solution, we report in the y-axis only the collected reward $\mathcal{R}(SOL)$ for each algorithm.

6.2. Results with a Single Drone

In this section, we evaluate our algorithms when solving MDMP with a single drone scenario. In Section 6.2.1, we assess the impact of drone altitude, while in Section 6.2.2, we assess the impact of energy and storage constraints.

6.2.1. Impact of the Altitude

In this section, we evaluate the impact of the altitude of the single drone.

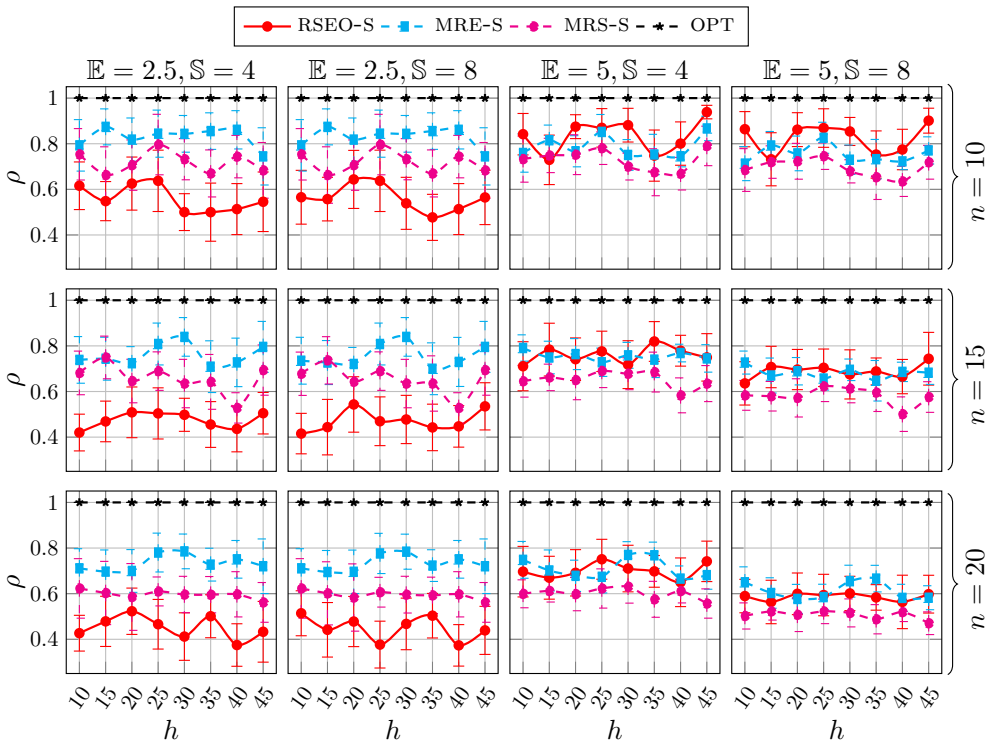

Figure 3: Single-drone: comparison of all the algorithms when varying the drone's altitude.

In Figure 3, we vary the altitude of the drone $h = \{10, \ldots, 45\}$ m in the x-axis, while we report the ratio ρ in the y-axis. We fixed small instances with $n = \{10, 15, 20\}$ sensors, since we also executed the OPT algorithm. In particular, since n is small, we have chosen small values for energy and storage constraints, i.e., $\mathbb{E} = \{2.5, 5\}$ MJ and $\mathbb{S} = \{4, 8\}$ GB, respectively, obtaining so 4 different combinations of energy-storage. The plots in Figure 3 are organized in 3 rows, i.e., the first row shows the result with $n = 10$, the second and the third show $n = 15$ and $n = 20$, respectively; while the 4 columns show the combinations of energy-storage mentioned above.

The first observation in Figure 3 is that, by fixing a particular energy-storage setting, the results slightly change when we vary the height of the drone, with a few exceptions. The altitude parameter affects the number of waypoints that the drone can consider. In fact, the higher is the drone's altitude, the less is the number of intersections among the sensors, and hence the less will be the actual number of waypoints. Regardless of the drone's height, we can see that the RSEO-S algorithm poorly performs when the energy budget is small ($\mathbb{E} = 2.5$ MJ). This is due to the fact that its initial strategy is aimed at finding a good partition of sensors such that the storage constraint is met. The selected sensors can belong to much different sub-areas of the field, and

therefore the energy required for visiting all of them could be not sufficient. When the energy is doubled ($\mathbb{E} = 5\,\text{MJ}$), however, the drone has much higher chances of finding a suitable route that can cover all the selected sensors and for this reason the RSEO-S algorithm performs better.

It is also interesting to see a counter-intuitive behavior with regard to the storage constraint. In fact, when the energy is small ($\mathbb{E} = 2.5\,\text{MJ}$), the more storage availability does not affect the results, neither positively nor negatively. This can be justified by the fact that when the drone has a limited battery, the resulted mission cannot be very long, and hence the larger availability of the storage is not detrimental. Instead, when the energy is larger ($\mathbb{E} = 5\,\text{MJ}$), a larger storage ($\mathbb{S} = 8\,\text{GB}$) slightly worsens the performance in terms of ratio. This is probably because the drone can visit more sensors flying longer routes, but the more availability of storage is not optimally exploited by the suboptimal algorithms.

The two greedy heuristic algorithms MRE-S and MRS-S perform more or less the same, with MRE-S that always outperforms MRS-S when varying the drone's height. When the energy is small ($\mathbb{E} = 2.5\,\text{MJ}$), the gap between the two strategies is more evident. In fact, MRE-S considers the energy budget when it builds the drone's mission. However, the time complexity of MRS-S is the lowest among all compared algorithms and often shows good enough performance.

In conclusion, since the height of the drone does not heavily affect the performance of our proposed algorithms, in the following, we consider a reasonable drone's height fixed to $h = 20\,\text{m}$.

6.2.2. Impact of the Energy and Storage Constraints

In this section, we evaluate the impact of energy and storage constraints for a given drone's altitude, i.e., $h = 20\,\text{m}$. In particular, we considered many more sensors in this comparison and therefore we do not execute the optimal OPT algorithm when $n \geq 25$.

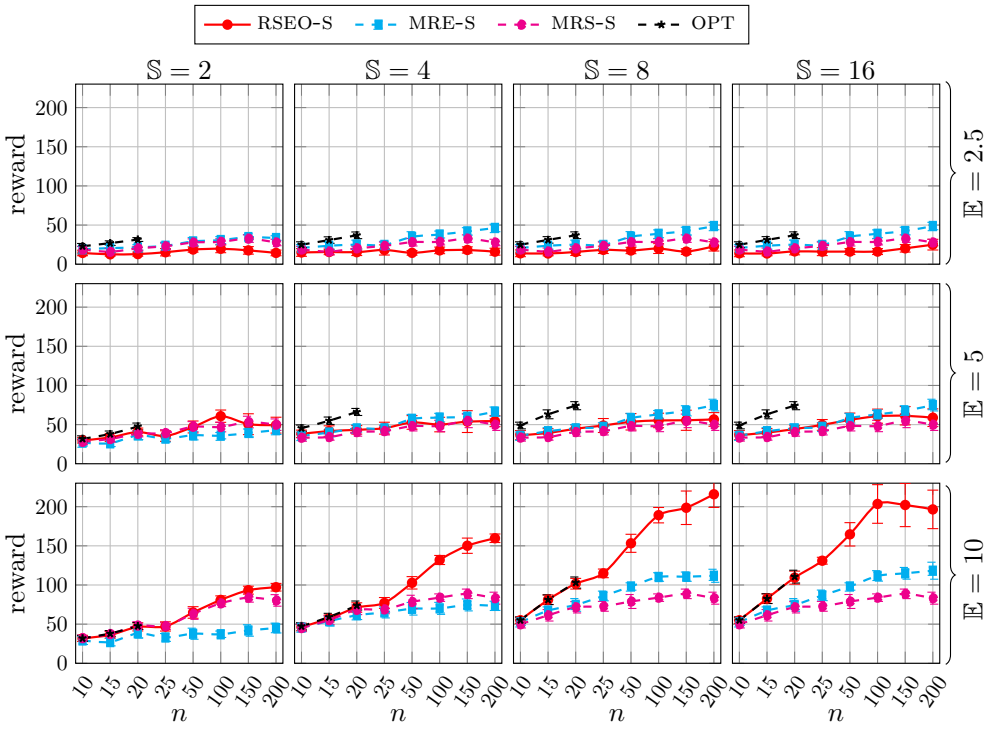

Figure 4: Single-drone: comparison of all the algorithms when varying the drone's energy and storage constraints.

In Figure 4, we fix the drone's altitude to $h = 20\,\text{m}$. We set the number of sensors $n = \{10, \ldots, 200\}$ on the x-axis, while we report the reward collected in the y-axis. Moreover, we set the drone's energy $\mathbb{E} = \{2.5, 5, 10\}\,\text{MJ}$ and storage $\mathbb{S} = \{2, 4, 8, 16\}\,\text{GB}$. The plots in Figure 4 are organized in 3 rows and 4 columns, i.e., fixing the energy values on rows, and the storage values on columns.

In Figure 4 we can observe how the energy influences the performance of the algorithms. In particular, it is interesting to see the behavior when the energy $\mathbb{E}$ changes from $2.5\,\text{MJ}$ to $10\,\text{MJ}$, specifically for $n = \{10, 15, 20\}$, i.e., the cases where we also performed the optimal OPT algorithm. Let us now focus on the case with $\mathbb{S} = 8\,\text{GB}$. In fact, when $\mathbb{E} = 2.5\,\text{MJ}$, the gap between the optimal algorithm and the other suboptimal ones is limited. Instead, when $\mathbb{E} = 5\,\text{MJ}$, such a gap is much more evident than before. However, when $\mathbb{E} = 10\,\text{MJ}$, the gap is reduced and, in addition, RSEO-S matches OPT. This is due to the fact that the RSEO-S algorithm can avoid pruning the sensors if the energy budget is very large.

Another interesting aspect to discuss is the collected reward when the number of sensors increases from $n = 100$ to $n = 200$. In fact, the total collected reward by the RSEO-S algorithm slightly decreases when $n = 200$. Recall that the strategy of RSEO-S is to consider the waypoints as a whole. Therefore, since the area of the field F is the same, when the number of sensors increases, the density of them in F also increases accordingly. As a consequence, the number of intersections dramatically increases and the hovering time on these will increase as well. Furthermore, during the pruning phase of RSEO-S, a certain number of waypoints, densely populated by sensors, will be discarded in order to keep the route within the energy constraint. Therefore, the final energy-feasible route will be sacrificed too much and there is a serious possibility that a residual energy budget can exist. In fact, we have experimentally observed that when $n = 200$, the residual non-used energy battery is a little bit larger than the cases when $n = 100$. This problem could be partially avoided by implementing a route reward-increasing phase in order to regain the unused energy, but this would not improve the guaranteed approximation ratio provided by the algorithm proved in Theorem 2.

In general, MRE-S outperforms MRS-S, especially when storage availability is high. Instead, for small storage in the input ($\mathbb{S} \leq 4\,\text{GB}$), the performance of MRS-S is better than that of MRE-S. Despite its relatively poor general performance, the MRS-S algorithm is still worthy due to its lower time complexity with respect to that of MRE-S.

6.3. Results with Multiple Drones

In this section, we evaluate our algorithms when solving MDMP with a multiple-drone scenario.

In Figure 5, we fix the drone's height to $h = 20\,\text{m}$. We set the number of sensors $n = \{10, \ldots, 200\}$ on the x-axis, while we report the collected reward on the y-axis. We set the drone's energy $\mathbb{E} = \{2.5, 5, 10\}\,\text{MJ}$ and the storage $\mathbb{S} = \{2, 4\}\,\text{GB}$. The small values of the storage are justified by the fact that here we employ a fleet of $l = \{2, 3, 4\}$ drones, and hence drones can have less capabilities. The plots in Figure 4 are organized in 3 rows and 4 columns, i.e., fixing the number of drones on rows, and the energy-storage combinations on columns. Due to the large number of constraints, the optimal OPT algorithms has been executed only for the smallest instances.

The general trend that we can observe in Figure 5 is that when the energy is very small ($\mathbb{E} = 2.5\,\text{MJ}$), the best performing algorithm is CAA-M, while when the energy increases to $\mathbb{E} = 5\,\text{MJ}$, the best algorithm is SAS-M (when the BFS visit is implemented), and finally when the energy increases to $\mathbb{E} = 10\,\text{MJ}$, the best algorithm is RSEO-M. With respect to SAS-M, we can see a symmetric behavior with respect to the energy in the input. In fact, when the energy is small, the

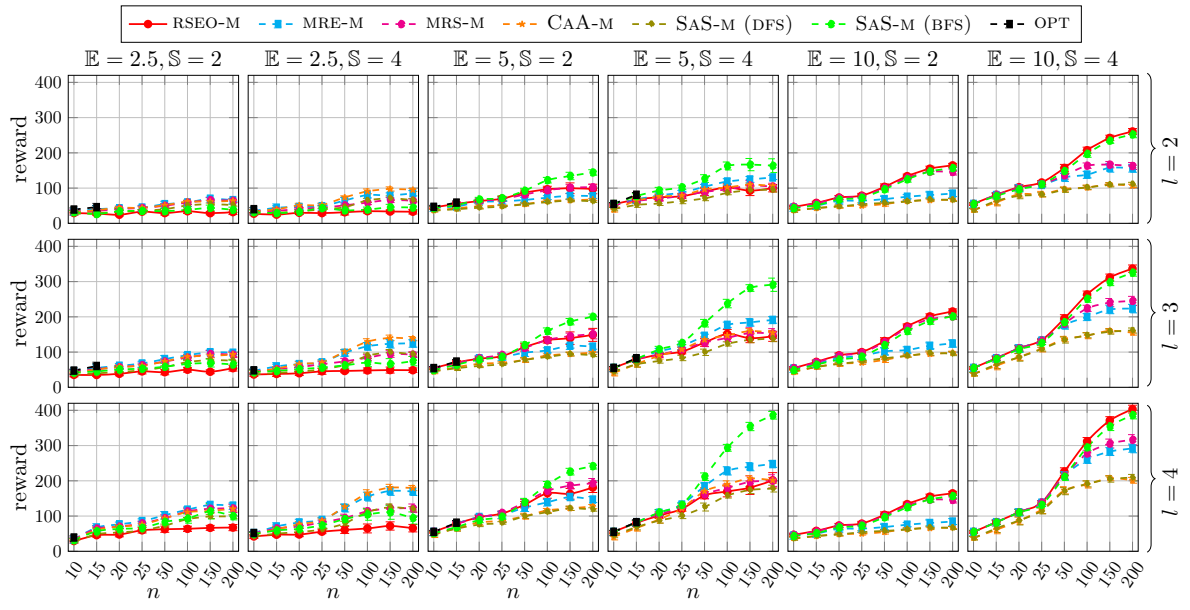

Figure 5: Multi-drone: comparison of all the algorithms when varying the drone's energy and storage constraints.

DFS version of SAS-M outperforms the BFS one, while with more energy it happens the opposite. In the latter case, the gap between the two visit strategies is more evident. We can also observe that the performance of SAS-M increases when the number of drones l also increases. Probably, this is justified by the fact that the DFS version tends to balance the distance of the sensors in each partition, whereas the BFS version is inclined to aggregate sensors into partitions with incremental distance from the depot. Therefore, when the budget is limited, the balanced nature of SAS-M DFS determines more profitable partitions, vice versa, it tends to "waste" energy by including very far sensors. The opposite reasoning can be applied for SAS-M BFS.

The RSEO-M algorithm's performance is poor when the energy level is low ($\mathbb{E} \leq 5\,\text{MJ}$). However, when there is more energy available ($\mathbb{E} = 10\,\text{MJ}$), the algorithm significantly improves. This trend is consistent with observations from the single drone scenario, where limited energy availability negatively affects the construction of an energy-feasible route for each drone. In contrast, if there is a large amount of energy available, the sub-sets of sensors returned through the *multi-knapsack* procedure become much easier to visit. Therefore, the algorithm's performance is dependent on the availability of energy, and it is more efficient when there is a higher energy level.

Finally, with respect to greedy heuristic algorithms, the MRE-M algorithm confirms to be a valid choice, although in the combination $\mathbb{E} = 10\,\text{MJ}, \mathbb{S} = 2\,\text{GB}$ its performance is not so good. This is because there is a high availability of energy, but a very low availability of storage. The other greedy solution, i.e., MRS-M, is almost always outperformed by MRE-M, but its time complexity is the lowest, and this is a good trade-off when the number of sensors and drones increases.

7. Conclusion

In this paper, we investigated the problem of using a fleet of drones to collect data from IoT ground sensors deployed in a field to be monitored. As an example, in a smart agriculture scenario, a fleet of drones is in charge of retrieving data from many sensors to detect the presence of insects

in orchards. The drones are constrained by both the available energy battery and the storage. The data that sensors have to offload to the drones is characterized by a size, and by a reward that models its relevance. The proposed problem is MDMP, whose goal is to plan a suitable set of missions for the drones such that the sum of the overall collected reward is maximized and the energy and storage constraints on each drone are both satisfied. We formally proved that MDMP is *NP*-hard, and presented an ILP formulation that optimally solves it. We also devised time-efficient approximation and heuristic algorithms capable of suboptimally solving MDMP.

In future work, we would like to incorporate communication issues between drones and sensors, as well as a more realistic environment with obstacles. Moreover, we can allow drones to fly at different altitudes during the same mission, for example, to cover more sensors or to improve air-to-ground communications. Finally, we plan to build a preliminary real test-bed with a single drone to collect data from real IoT sensors that store text and image information.

References

[1] F. Betti Sorbelli, A. Navarra, L. Palazzetti, C. M. Pinotti, G. Prencipe, Optimal and heuristic algorithms for data collection by using an energy-and storage-constrained drone, in: International Symposium on Algorithms and Experiments for Wireless Sensor Networks, Springer International Publishing, Cham, 2022, pp. 18–30.

[2] T. Calamoneri, F. Corò, S. Mancini, A Realistic Model to Support Rescue Operations after an Earthquake via UAVs, IEEE Access (2022).

[3] A. Shabani, B. Asgarian, S. A. Gharebaghi, M. A. Salido, A. Giret, A new optimization algorithm based on search and rescue operations, Mathematical Problems in Engineering 2019 (2019).

[4] F. Betti Sorbelli, F. Corò, S. K. Das, L. Palazzetti, C. M. Pinotti, On the scheduling of conflictual deliveries in a last-mile delivery scenario with truck-carried drones, Pervasive and Mobile Computing 87 (2022) 101700.

[5] J.-P. Aurambout, K. Gkoumas, B. Ciuffo, Last mile delivery by drones: An estimation of viable market potential and access to citizens across european cities, European Transport Research Review 11 (1) (2019) 1–21.

[6] F. Betti Sorbelli, C. M. Pinotti, G. Rigoni, On the evaluation of a drone-based delivery system on a mixed euclidean-manhattan grid, IEEE Transactions on Intelligent Transportation Systems (2022).

[7] P. P. Roosjen, B. Kellenberger, L. Kooistra, D. R. Green, J. Fahrentrapp, Deep learning for automated detection of drosophila suzukii: potential for uav-based monitoring, Pest Management Science 76 (9) (2020) 2994–3002.

[8] F. Betti Sorbelli, F. Corò, S. K. Das, L. Palazzetti, C. M. Pinotti, Drone-based optimal and heuristic orienteering algorithms towards bug detection in orchards, in: 2022 18th International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE, 2022, pp. 117–124.

[9] HALY.ID, Project, `https://www.haly-id.eu` (2022).

[10] F. Betti Sorbelli, S. Carpin, F. Corò, S. K. Das, A. Navarra, C. M. Pinotti, Speeding up routing schedules on aisle graphs with single access, IEEE Transactions on Robotics (2021).

[11] S. Petkovic, D. Petkovic, A. Petkovic, Iot devices vs. drones for data collection in agriculture, DAAAM International Scientific Book 16 (2017) 63–80.

[12] C. Caillouet, F. Giroire, T. Razafindralambo, Efficient data collection and tracking with flying drones, Ad Hoc Networks 89 (2019) 35–46.

[13] E. Aras, M. Ammar, F. Yang, W. Joosen, D. Hughes, Microvault: Reliable storage unit for iot devices, in: 2020 16th International Conference on Distributed Computing in Sensor Systems (DCOSS), IEEE, 2020, pp. 132–140.

[14] H. Wang, H. Ke, W. Sun, Unmanned-aerial-vehicle-assisted computation offloading for mobile edge computing based on deep reinforcement learning, IEEE Access 8 (2020) 180784–180798.

[15] X. He, R. Jin, H. Dai, Multi-hop task offloading with on-the-fly computation for multi-uav remote edge computing, IEEE Transactions on Communications (2021).

[16] G. D'Angelo, D. Diodati, A. Navarra, C. M. Pinotti, The minimum k-storage problem: Complexity, approximation, and experimental analysis, IEEE Trans. on Mobile Comp. 15 (7) (2015).

[17] M. Chen, W. Liang, Y. Li, Data collection maximization for uav-enabled wireless sensor networks, in: 29th Intl. Conf. on Computer Communications and Networks (ICCCN), IEEE, 2020, pp. 1–9.

[18] M. Chen, W. Liang, J. Li, Energy-efficient data collection maximization for uav-assisted wireless sensor networks, in: Wireless Communications and Networking Conf. (WCNC), IEEE, 2021, pp. 1–7.

[19] M. Chen, W. Liang, S. K. Das, Data collection utility maximization in wireless sensor networks via efficient determination of uav hovering locations, in: PerCom, IEEE, 2021, pp. 1–10.
[20] J. Zhang, Z. Li, W. Xu, J. Peng, W. Liang, Z. Xu, X. Ren, X. Jia, Minimizing the number of deployed uavs for delay-bounded data collection of iot devices, in: INFOCOM, IEEE, 2021, pp. 1–10.
[21] C. Zhan, Y. Zeng, R. Zhang, Energy-efficient data collection in uav enabled wireless sensor network, IEEE Wireless Communications Letters 7 (3) (2017) 328–331.
[22] R. A. Nazib, S. Moh, Energy-efficient and fast data collection in uav-aided wireless sensor networks for hilly terrains, IEEE Access 9 (2021) 23168–23190.
[23] J. Liu, P. Tong, X. Wang, B. Bai, H. Dai, Uav-aided data collection for information freshness in wireless sensor networks, IEEE Trans. on Wireless Communications 20 (4) (2020) 2368–2382.
[24] X. Li, J. Tan, et al., A novel uav-enabled data collection scheme for intelligent transportation system through uav speed control, IEEE Trans. on Intelligent Transportation Systems 22 (4) (2020).
[25] M. B. Ghorbel, D. Rodriguez-Duarte, H. Ghazzai, M. J. Hossain, H. Menouar, Energy efficient data collection for wireless sensors using drones, in: 2018 IEEE 87th Vehicular Technology Conference (VTC Spring), IEEE, 2018, pp. 1–5.
[26] R. I. da Silva, M. A. Nascimento, On best drone tour plans for data collection in wireless sensor network, in: Proceedings of the 31st annual ACM symposium on applied computing, 2016, pp. 703–708.
[27] C. Pu, A. Wall, I. Ahmed, K.-K. R. Choo, Secureiod: A secure data collection and storage mechanism for internet of drones, in: 2022 23rd IEEE International Conference on Mobile Data Management (MDM), IEEE, 2022, pp. 83–92.
[28] A. Trotta, M. Di Felice, L. Bononi, E. Natalizio, L. Perilli, E. F. Scarselli, T. S. Cinotti, R. Canegallo, Beedrones: Energy-efficient data collection on wake-up radio-based wireless sensor networks, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2019, pp. 547–553.
[29] D. Zorbas, B. O'Flynn, Collision-free sensor data collection using lorawan and drones, in: 2018 Global Information Infrastructure and Networking Symposium (GIIS), IEEE, 2018, pp. 1–5.
[30] R. I. da Silva, J. D. C. V. Rezende, M. J. F. Souza, Collecting large volume data from wireless sensor network by drone, Ad Hoc Networks 138 (2023) 103017.
[31] I.-M. Chao, B. L. Golden, E. A. Wasil, The team orienteering problem, European journal of operational research 88 (3) (1996) 464–474.
[32] P. Vansteenwegen, et al., The orienteering problem: A survey, European Journal of Op. Research 209 (1) (2011).
[33] V. V. Vazirani, Approximation algorithms, Vol. 1, Springer, 2001.
[34] S. Martello, P. Toth, Solution of the zero-one multiple knapsack problem, European Journal of Operational Research 4 (4) (1980) 276–283.
[35] C. Chekuri, S. Khanna, A polynomial time approximation scheme for the multiple knapsack problem, SIAM Journal on Computing 35 (3) (2005) 713–728.
[36] K. Jansen, A fast approximation scheme for the multiple knapsack problem, in: International Conference on Current Trends in Theory and Practice of Computer Science, Springer, 2012, pp. 313–324.
[37] S. Martello, P. Toth, Heuristic algorithms for the multiple knapsack problem, Computing 27 (2) (1981) 93–112.
[38] S. Martello, P. Toth, Knapsack problems: algorithms and computer implementations, John Wiley & Sons, Inc., 1990.
[39] J. Kleinberg, E. Tardos, Algorithm design, Pearson Education India, 2006.
[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2022.
[41] J. Sreevalsan-Nair, K-means clustering, in: Encyclopedia of Mathematical Geosciences, Springer, 2021, pp. 1–3.
[42] F. Nie, Z. Li, R. Wang, X. Li, An effective and efficient algorithm for k-means clustering with new formulation, IEEE Transactions on Knowledge and Data Engineering (2022).
[43] J. Jansons, T. Dorins, Analyzing ieee 802.11 n standard: outdoor performanace, in: 2012 Second International Conference on Digital Information Processing and Communications (ICDIPC), IEEE, 2012, pp. 26–30.
[44] 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 communications 9 (1) (2018) 1–13.
[45] A. Khochare, Y. Simmhan, F. Betti Sorbelli, S. K. Das, Heuristic algorithms for co-scheduling of edge analytics and routes for uav fleet missions, in: IEEE INFOCOM 2021-IEEE Conference on Computer Communications, IEEE, 2021, pp. 1–10.