Received December 16, 2021, accepted January 3, 2022, date of publication January 7, 2022, date of current version January 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141216

A Realistic Model to Support Rescue Operations After an Earthquake via UAVs

TIZIANA CALAMONERI[1], FEDERICO CORÒ[2], (Member, IEEE), AND SIMONA MANCINI[3,4]
[1]Department of Computer Science, Sapienza University of Rome, 00161 Rome, Italy
[2]Department of Computer Science, Missouri University of Science and Technology, Rolla, MO 65409, USA
[3]Department of Science, Innovation and Technology, University of Eastern Piedmont, 15121 Alessandria, Italy
[4]Department of Operations, Energy, and Environmental Management, University of Klagenfurt, 9020 Klagenfurt, Austria

Corresponding author: Federico Corò (federico.coro@mst.edu)

This work was supported in part by the Sapienza University of Rome through the project Exploiting Hybrid Networks for Saving People after a Natural Disaster under Project RM11715C44AB9878.

ABSTRACT In this paper, we consider the problem of completely flying over an area just hit by an earthquake with a fleet of Unmanned Aerial Vehicles (UAVs) to opportunely direct rescue teams. The cooperation between UAVs ensures that the search for possible survivors can be faster and more effective than the solutions currently implemented by civil protection. To study this scenario, we introduce the Cover by Multitrips with Priorities (CMP) problem, which tries to keep into account all the main real-life issues connected to the flight and coordination of the UAVs. We conduct a theoretical study to estimate the best number of UAVs and additional batteries, to give indications to the organization that leads the rescue teams to be able to guarantee rapid and effective rescue. Finally, based on some theoretical considerations, we propose some heuristics that tackle the problem of flying over the whole area with a fleet of UAVs in the shortest possible time. Simulations show that they work efficiently in both the proposed scenarios and provide better performance than previous solutions once they are arranged to work in our scenarios. The main advantages of our approach w.r.t. the current drone-based solutions used by the civil defense are that UAVs do not need drivers so the time of all available rescue workers can be invested in doing something else. In our model, we take into account that some sites (*e.g.* buildings with a high fire risk or schools and hospitals) have a higher priority and must be inspected first, and the possibility that UAVs can make a decision based on what they detect. Finally, our approach allows UAVs to collaborate so that the same sites will be flown over exactly once in order to speed up the rescue mission.

INDEX TERMS Unmanned aerial vehicle networks, battery-aware cycle covering, UAV routing problem.

I. INTRODUCTION

In case of natural disasters, such as earthquakes, rescue teams must complete their operations within a few hours of the event to avoid increased loss of life. Furthermore, in the aftermath of these disasters, we cannot count on the presence of infrastructure (*e.g.*, there is no guarantee that internet will be available), and there may not be time to re-establish it to undertake coordinated activities. In this context, Unmanned Aerial Vehicles (UAVs) can make a difference to ensure rapid and effective aid. We recognize that the current legislation of several countries is not ready to immediately accept the implementation of solutions exploiting UAVs [1]. Nevertheless, civil defense, policies, and lawmakers are positively responding to the tremendous advantages given by innovative solutions based on the cooperation capabilities of UAVs [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Halil Ersin Soken.

Researchers strongly believe that, as we have already seen for artificial intelligence and robotic systems in many fields, UAVs will gain the trust of common people and lawmakers. This is also the reason why, with recent advances in UAV technologies, many papers have been published proposing different theoretical models for handling many algorithmic optimization problems related to UAV rescue operations. Nevertheless, since the problems related to these real-life issues are very complex, most of such models appear too simplified and assume some hypotheses that are not reasonable in practice. For example, in the majority of the papers dealing with Search and Rescue (*e.g.*, [3], [4]), UAVs are supposed not to have battery constraints at all, while these are usually rather pressing; in other cases, UAVs are assumed to be able to fly back to the base in a very short time (*e.g.* in [5] UAVs go to recharge their battery when they are at less than 5 minutes out of 28 of battery life left) but there is no certainty that this time will be enough; in [3], [6], [7] mobile phone signals are exploited to localize a person, but it is not certain that people keep their cell phones close to them when they are at home, especially at night; in [8], [9] UAVs send stream videos to the base station through a cellular network in mountain environments, where usually there is no guarantee of coverage; often UAVs have to fly over an entire area and not focus on certain buildings, making it impossible to assign priorities to certain places [10], [11]; finally, the costs that a UAV will face during its trip are usually considered as fully known [12]–[14], while in real-life scenarios, a UAV could spend a time even very different from the expected one to explore or traverse an area.

This paper addresses the problem of completely flying over an area just hit by an earthquake with a fleet of UAVs to opportunely direct rescue teams. The main advantages of our approach w.r.t. the current drone-based solutions used by the civil defense are that UAVs do not need drivers, so the time of all available rescue workers can be better invested; moreover, we take into account that some sites (*e.g.* buildings with a high fire risk or schools and hospitals) have a higher priority and must be inspected first. Finally, in order to speed up the rescue mission, our approach allows UAVs to collaborate so that the same sites will be flown over exactly once and consider the possibility that UAVs make a decision based on what they detect. This is important in order to have fast and efficient rescue operations.

More in detail, our main contributions are the following:

- we define a new problem, called Cover by Multitrips with Priorities (CMP), that tries to keep into account all the main real-life issues connected to the flight and coordination of the UAVs;
- we provide a complete graph model for it in two different realistic scenarios;
- we make a theoretical study to provide an *a priori* estimate on the completion time of the rescue operations and to estimate the best number of UAVs and additional batteries that guarantee a fast and effective rescue to the organization leading the rescue teams;
- we propose a mixed-integer linear programming formulation of our graph model;
- based on some theoretical considerations, we propose some heuristics that address the problem of flying over the whole area with a fleet of UAVs in the shortest possible time;
- we extend the algorithm proposed by Kim *et al.* [4], [15], which is one of the most complete solutions known in the literature, to make it work under the same operative scenarios addressed by our heuristics; we call this extension $\mathcal{A}_{TSPN}$;
- we perform extensive simulations that show that our heuristics work efficiently in both the proposed scenarios; furthermore, we experimentally compare our heuristics with $\mathcal{A}_{TSPN}$ and we show that our heuristics significantly outperform it in all the key performance metrics.

This paper is organized as follows. In the next section we describe our problem and compare it with some routing problems already studied in the literature that can appear somehow similar; we detail two possible real-life scenarios; for each, we propose the associated graph model, the adopted performance metrics, and a mixed-integer linear programming formulation. Section III is devoted to the description of a meta-algorithm that we exploit to design new heuristics.

In Section IV, we consider the possibility of replacing exhausted batteries with already loaded ones: we associate each possible value of the number of additional batteries with the corresponding completion time so that the civil defense organizers can have an immediate idea of the time needed to complete the operations. Knowing all the parameters, the minimum number of batteries necessary to guarantee the minimum completion time is also obtained (*i.e.* we impose to have no time in which the UAVs remain inactive waiting for their battery to be recharged).

In Section V we study the possibility of concluding the overflight of the area within a single flight (that is, without the need to recharge batteries) and we propose an algorithm to exactly determine the number of necessary UAVs.

Section VI is devoted to the design of five heuristics that we experimentally compare in Section VII.

Section VIII concludes the paper with a list of open problems.

II. THE PROBLEM: FROM REAL LIFE SITUATION TO MODELS

In this section, we first describe the real-life situation that we want to model in two possible scenarios, then we detail our corresponding graph problem, including a discussion on some useful optimization functions, and finally, we provide a mixed-integer linear programming formulation of it.

A. REAL LIFE SITUATION

We consider the very critical situation after a natural disaster (*e.g.* an earthquake) and the main problem of flying over the entire area using a fleet of UAVs equipped with sensors and/or cameras, acquiring information as soon as possible to have a clear idea of where rescue teams are most needed.

It is reasonable to assume that civil defense has an updated map of each area considered at risk and, for every relevant building, has an estimate of the information on the time needed to fly over it entirely and the level of importance. With the latter, we want to prioritize the most critical buildings, such as hospitals and schools or buildings with a high fire risk.

We consider to have a fleet of q homogeneous UAVs $u_1, \ldots, u_q$ that can work independently, taking off from the operations center v_0. Each UAV has a battery corresponding to $B \in \mathbb{R}^+$ units of flying time; when the battery is discharged, it takes $R \in \mathbb{R}^+$ time units to be fully recharged, and this can only be done at v_0. Technology currently sets $1.5B \leq R \leq 2B$, see for example [16], so it is suitable to assume $R \geq B$.

It is worth noting that, instead of a fleet of homogeneous UAVs, we could consider also heterogeneous devices, each one with a different battery constraint B_i and different cruise speeds and endurance. In this case, handling parameters becomes more complicated (*e.g.*, not necessarily battery units would correspond to units of flying time). For the sake of simplicity, we avoid this case but we discuss it in Subsection II-E how to change the MILP formulation in order to adapt it to this more general situation.

Since our scenario is a time-critical situation, we allow having a certain number $b \geq 0$ of batteries that can be recharged separately while the UAVs are in flight and can replace the discharged ones in a few seconds, without causing a delay in the takeoff for the next flight. Indeed, already in 2015, Lee *et al.* [17] used a small robot arm to do the battery swap, the whole process taking about 60 seconds; more recently, the authors of [18], [19] propose a low-cost system only taking 10 seconds to finish the battery swapping process, from landing to take off.

We explicitly avoid handling the collision problem (which is not the focus of this document and for which there is a large literature [20]). Connected to this, we assume that the UAVs are not equipped with infrared cameras, in fact, thermal imaging cameras have proven to be reliable in detecting humans over short distances (hence low and fixed altitude is required, making it more difficult to avoid collisions), but it results in an ambiguous blob formation when used from long distances or in the presence of numerous obstacles [21].

We consider two scenarios, depending on whether the UAV fleet consists of a set of small-scale UAVs or some high-cost UAVs that can be equipped with additional sensors. These represent the two extremes among the many realistic possibilities: on the one hand we assume that the rescue teams have many cheap UAVs that cannot take decisions, on the other hand, they may have a lower number of more powerful UAVs, that can both communicate with the base and take decisions.

In the *first scenario*, each UAV is equipped with a commercial RGB camera and has very small computing power and no communication devices. In this case, the q UAVs can only follow an assigned path through some sensible targets, take a video, and bring it back to the operations center where image detection/recognition tools will be able to detect, in parallel, possible survivors needing help in real-time. The positive side of this scenario is that the fleet can easily consist of a large number of UAVs due to their low cost.

In the *second scenario*, we consider high-cost UAVs, which have a processing unit with good computing power and can be equipped with a device that allows them to communicate with the base at any time (*e.g.* a radio); therefore they can produce and scan the videos in real-time(see *e.g.* [22]–[24]) in order to immediately detect and recognize if there is an emergency; if people needing help are detected, the UAVs will promptly get closer to possible survivors, to acquire more information and send a message to the base for a rescue team to be sent.

It should be noted that, due to legal constraints, both proposed scenarios are more advanced than those currently adopted by the civil defense, which either uses human-guided drones and/or allows UAVs to fly only by sight from the base.

B. RELATED PROBLEMS

Before detailing the graph model, the performance metrics, and the MILP formulation for our problem, we briefly discuss here similarities and differences between our scenarios and some routing problems already studied in the literature.

Although the problem addressed in this paper deals with limited autonomy of the vehicles due to battery capacity, it is deeply different from the routing problems involving *e.g.* electric vehicles, which have been broadly analysed in the literature [25]. In fact, those problems allow in-route battery recharging (or swapping) at specific locations (recharging stations) spread across the territory. Instead, in our case, batteries can be swapped only at the depot and not during the trip's execution.

A problem that could seem close to ours is the multi-trip vehicle routing problem (MTVRP) [26], in which vehicles, whose trips are not limited by the driving range but by the load capacity, may perform several trips in sequence. At the end of each trip, the vehicle returns to the depot in order to load other goods and perform the next trip. Generally, a maximum cumulative working time is imposed for each vehicle. The main difference between MTVRP and our problem concerns the objective function. In fact, while in MTVRP the goal is to minimize covered distances, in our case, we look for weighted completion time minimization. Moreover, MTVRP applications are mainly related to freight delivery (or pick-up). In such a context, the minimization of costs, which are strictly dependent on the traveled distance, is the goal of the company. In our setting, consisting in monitoring an area hit by a natural disaster (such as an earthquake), the primary objective is to serve the points of interest in the shortest possible time, considering that some sites must be flown over before than others. Indeed, most densely populated areas and some crucial buildings are marked with a higher priority because the probability of finding injured people is larger in these zones. Although completion time minimization has been already considered in the literature for a single vehicle problem [27], we are the first researchers to apply it in a multi-trip and multi-vehicle context.

C. THE GRAPH MODEL

In order to model the described context, we consider a complete node- and edge-weighted graph $G = (V, E, dist, \sigma, p)$, where:

- $V = \{v_0, v_1, \ldots, v_n\}$, and $v_1, \ldots, v_n$ are the sites representing the points of interest inside the area, *i.e.*, buildings (or areas) that need to be explored, while v_0 represents the operations center.
- Functions $p : V \rightarrow \{p_{min}, p_{med}, p_{max}\}$, with $p_{min} < p_{med} < p_{max}$, and $\sigma : V \rightarrow \mathbb{R}^+$ associate to each node two values, representing the priority and the time needed to

$u_1, \dots u_q$	fleet of q homogeneous UAVs;
v_0	operations center;
$B \in \mathbb{R}^+$	battery in terms of units of flying time;
$b \in \mathbb{N}$	number of additional batteries;
$R \in \mathbb{R}^+$	recharging time; $R \geq B$;
$V = \{v_1, ..., v_n\}$	set of n sites;
$p : V \to \{p_{min}, p_{med}, p_{max}\}$	priority of each site;
$\sigma : V \to \mathbb{R}^+$	time needed to explore each site; known as part of the input;
$\sigma' : V \to \mathbb{R}^+$	(only in the second scenario) additional and *a priori* unknown time needed to explore in deeper detail each site;
$\sigma^e : V \to \mathbb{R}^+$	$\sigma^e(v) = \sigma(v) + \sigma'(v)$;
$dist : E \to \mathbb{R}^+$	symmetric traveling distance between the extremes of each edge;
$l : E \to \mathbb{R}^+$	$l(v_i, v_j) = dist(v_i, v_j) + \frac{\sigma(v_i)}{2} + \frac{\sigma(v_j)}{2}$ for each $i, j \neq 0$; it holds $2l(v_0, v) + \sigma'(v) \leq B$;
$G = (V, E, dist, \sigma, p)$	complete graph on node set V with edge-weight function $dist$ (see Subsec. II-C);
$C = \langle v_0, v_{i_1}, \dots, v_{i_m}, v_0 \rangle$	(presumed) cycle that a UAV can theoretically overfly in a single flight (*i.e.* considering $\sigma' = 0$) (see Def. 1);
$C^e = \langle v_0, v_{i_1}, \dots, v_{i_{m'}}, v_0 \rangle$	effective cycle associated to a presumed one (see Def. 2);
$t_s(C)$ and $t_f(C)$	starting and finishing time of cycle C (either presumed or effective);
$t(C) = t_f(C) - t_s(C)$	overflight time of cycle C (either presumed or effective);
$\mathcal{S}_{u_i} = \{C_1(u_i), \dots, C_{y_i}(u_i)\}$	sequence, *i.e.* ordered set of cycles associated to UAV u_i (see Def. 3);
SOL(SOL^e)	collection of presumed (effective) sequences, one for each UAV, whose union covers V (see Def. 3);
$cost^{(i)}(v_k, \text{SOL}), i = I, II$	completion time of site v_k w.r.t. solution SOL in scenario i (see Def. 4);
$N(\text{SOL})$ (for short N)	number of cycles in SOL;
$wL^{(i)}(\text{SOL}^e), i = I, II$	weighted latency of SOL^e in scenario i (see Def. 5);
$ct(\text{SOL}^e)^{(i)}, i = I, II$	completion time of SOL^e in scenario i (see Def. 6);
CMP	Cover by Multitrips with Priorities Problem (see Def. 7);

FIGURE 1. **List of symbols used along the paper.**

explore the site, respectively. The value of $\sigma(v)$ is computed based on the shape and dimension of the target and is given as part of the input (We do not further detail how σ is computed because it is out of the scope of this paper, and simply assume that it is a known value.) In the first scenario, it represents the only (fixed) time to fly over each site. In the second scenario, $\sigma(v)$ includes both the time necessary to fly over the site and to process the images; moreover, it may be necessary to add a second contribution to $\sigma(v)$, *i.e.* the time $\sigma'(v)$ needed for the UAV to decrease its altitude, take more precise information concerning detected people possibly in difficulty and send to the base a radio message. We will denote by $\sigma^e(v)$ the effective time used by a UAV to completely fly over a site, *i.e.* $\sigma^e(v) = \sigma(v) + \sigma'(v)$ and $\sigma^e(v)$ becomes known only after that a UAV has finished to fly over v. As far as the effective overflight time is conceived, it always holds $\sigma(v) \leq \sigma^e(v)$; indeed, if no people needing help are detected, $\sigma^e(v) = \sigma(v)$; on the contrary, if a rescue team is deemed necessary, $\sigma^e(v) = \sigma(v) + \sigma'(v)$ with $\sigma'(v) > 0$.

In both scenarios, it is not restrictive to define $\sigma(v_0) = 0$; indeed, even if the base had to have a not negligible area, and each UAV had to take a time s to fly over it and reach the destination, this would be equivalent to set $\sigma(v_0) = 0$ and battery constraint equal to $B - s$.

• Every (directed) edge $e = (v_i, v_j) \in E$, with $i \neq j$, is associated a *distance* function *dist* that represents the traveling distance between two nodes.

Moreover, we assume that for *dist* the triangle inequality holds. For the feasibility of the problem, we need that each site is reachable and completely flown over by a UAV, *i.e.* for each site $v \in V$ it must hold $2\,dist(v_0, v) + \sigma(v) + \sigma'(v) \leq B$. If there is a very large site v that is reachable but takes a too long time to fly over (so that $2\,dist(v_0, v) + \sigma(v) + \sigma'(v) > B$), we replace v with a set $v^1, \dots, v^{m_v}$ of m_v nodes, so that the overflying time is split among the duplicates: $\sigma(v^i) = \frac{\sigma(v)}{m_v}$ for each $i = 1, \dots, m_v$; these m_v sites are considered as different sites at null mutual distance and they keep the same set of adjacent nodes as v. The value of m_v must be enough to guarantee that every v_i is reachable and completely flown over by a UAV; in other words, m_v is such that, for every v^i, it holds $2\,dist(v_0, v^i) + \sigma(v^i) + \sigma'(v^i) \leq B$, $i = 1, \dots, m_v$. Trivially, $p(v^i) = p(v)$, $i = 1, \dots, m_v$ and it is not necessary to highlight which nodes in the input graph are of this kind. Hence, in the rest of this paper, we implicitly assume that all the edges respect the feasibility requirement.

Since we can translate the value of *dist* in terms of flight time, as in [28], [29], we assign to each node an edge weight function $l : E \to \mathbb{R}^+$ defined as $l(v_i, v_j) = dist(v_i, v_j) + \frac{\sigma(v_i)}{2} + \frac{\sigma(v_j)}{2}$ for each $i, j \neq 0$. The triangle inequality still holds for l. In this way, from now on we will handle the only edge weight function l instead of *dist* and σ.

The goal of our problem is to inspect all points of interest in the shortest possible time, trying to explore sites with higher priority first; to this aim, we partition the sites into disjoint sets and assign each UAV one of such sets to fly over, so that the entire fleet can cooperate to the complete exploration, and avoid that some sites are flown over by more than one UAV.

Since UAVs are prone to their battery constraint, they may not be able to visit all assigned sites in one flight but may need to perform multiple flights, in order to either recharge or substitute their batteries. So, we have to assign to each UAV an ordered set of cycles, whose weights are bounded by B, and all nodes of the graph are flown over, giving precedence to those with higher priority, in such a way that an appropriate optimization function is minimized. Of course, in the two scenarios, the situation is different and hence also the solution is. In particular, it is worth noting that, while in the first scenario the time it takes to explore each site is exact (*i.e.* $\sigma'(v) = 0 \; \forall v$), in the second one it is not (and hence $\sigma'(v)$ can be either positive or null). The optimization problem is not run on-board but on a dedicated computer at the operations center in order not to require the usage of wireless connectivity among UAVs.

To consider this difference and, at the same time, to give definitions suitable for both scenarios, we introduce the concepts of presumed and effective cycles.

Informally, a presumed cycle is a cycle that can be flown over in a time-bounded by B assuming that, for each node involved in the cycle, the value of σ' is null; after this cycle is assigned to a UAV and it begins to fly over it, if some values of σ' are not null, it may become impossible to complete the exploration of the cycle within time B; so only a proper subset of sites involved in the presumed cycle will be really flown over, and the UAV will have to go back to the base: the flown over part of the presumed cycle is the effective cycle. Trivially, in the first scenario presumed and effective cycles coincide, while in the second scenario they can be different. We now formalize these definitions and state the properties of a feasible solution and the objective functions in the two scenarios.

Definition 1: A *(presumed) cycle* is a tuple $C = \langle v_0, v_{i_1}, \ldots, v_{i_m}, v_0 \rangle$ of sites that a UAV can explore in this order in a single flight, assuming $\sigma'(v) = 0 \; \forall v \in C$.

We fix time 0 at the beginning of operations for the whole system; denoting by $t_s(C)$ and $t_f(C)$ the *starting* and *finishing times of* each *cycle* C, and setting $i_0 = i_{m+1} = 0$, in both scenarios it holds that

$$t(C) = t_f(C) - t_s(C) = \sum_{j=0}^{m} l(v_{i_j}, v_{i_{j+1}}) \leq B. \tag{1}$$

This inequality is not guaranteed to be true, when passing to the effective times, *i.e.* for

$$l(v_0, v_{i_1}) + \sum_{j=1}^{m} \left(l(v_{i_j}, v_{i_{j+1}}) + \sigma'(v_{i_j}) \right). \tag{2}$$

We hence introduce the following:

Definition 2: The *effective cycle* associated to a presumed cycle $C = \langle v_0, v_{i_1}, \ldots, v_{i_m}, v_0 \rangle$ is a cycle containing a subset of sites of C: $C^e = \langle v_0, v_{i_1}, \ldots, v_{i_{m'}}, v_0 \rangle$ where $m' \leq m$, and m' is maximum to guarantee that

$$l(v_0, v_{i_1}) + \sum_{j=1}^{m'} \left(l(v_{i_j}, v_{i_{j+1}}) + \sigma'(v_{i_j}) \right) \leq B. \tag{3}$$

We define $t_s(C^e)$, $t_f(C^e)$ and $t(C^e)$ analogously to $t_s(C)$; $t_f(C)$ and $t(C)$. (In general, if C^e is the effective cycle associated to C, $t_s(C) = t_s(C^e)$ but $t_f(C) \neq t_f(C^e)$ and $t(C) \neq t(C^e)$.)

Definition 3: The *presumed (effective) sequence* associated to a UAV u_i, $\mathcal{S}_{u_i} = \{C_1(u_i), \ldots, C_{y_i}(u_i)\}$ ($\mathcal{S}^e_{u_i} = \{C^e_1(u_i), \ldots, C^e_{y_i}(u_i)\}$) is defined as an ordered set of presumed (effective) cycles assigned to UAV u_i.

A set SOL (SOL^e), defined as a collection of presumed (effective) sequences, one for each UAV, is called *presumed* (*effective*) *solution* if it covers V, *i.e.* if for every $v \in V$ there exists a cycle in a sequence of SOL (SOL^e) containing v.

The final result we would like to obtain is an effective solution SOL^e. In particular, in the first scenario, we are able to compute *a priori* a set of (presumed = effective) sequences that will take part in the (presumed = effective) solution. On the contrary, in the second scenario, we will not compute all cycles of a sequence at the same time: as we will detail in Section III, we will compute a presumed cycle of a sequence only after that the corresponding UAV is back from the effective previous cycle.

D. PEFORMANCE METRICS

We want the completion time of a site to represent the time necessary to know whether there are people needing help on it. In the two scenarios, we consider a site as "served" by a solution SOL if different conditions are verified. Namely, in the first scenario, a node is "served" only after that the video of the cycle including it has been delivered to the base and the portion corresponding to it has been analyzed, while in the second scenario a node is "served" as soon as it has been completely flown over since UAVs can immediately detect people needing help. We formalize these facts in the following:

Definition 4: Given a solution SOL^e and assuming that site v_k is the k-th in the effective cycle C^e (belonging to a sequence of SOL^e), the *completion time of* v_k *in a solution* SOL^e in the first scenario is:

$$cost^{(I)}(v_k, \mathrm{SOL}^e) = t_f(C^e) + t^{(I)}(v_k) \tag{4}$$

where

$$t^{(I)}(v_k) = \sum_{j=0}^{k-1} l(v_{i_j}, v_{i_{j+1}}) + \frac{\sigma(v_k)}{2}; \tag{5}$$

while in the second scenario is:

$$cost^{(II)}(v_k, \mathrm{SOL}^e) = t_s(C^e) + t^{(II)}(v_k) \tag{6}$$

where

$$t^{(II)}(v_k) = \sum_{j=0}^{k-1} l(v_{i_j}, v_{i_{j+1}}) + \frac{\sigma(v_k)}{2} + \sum_{j=1}^{k} \sigma'(v_j); \tag{7}$$

Note that the first term of $cost^{(I)}$ is the time necessary to completely fly over the cycle containing v_k (C^e) and the second term is the time necessary to analyze the recorded video from the beginning to the moment when v_k has been flown over. On the contrary, the first term of $cost^{(II)}$ is the time spent before starting flying over C^e and the second term is the time necessary to fly over the portion of C^e preceding v_k and v_k itself (including both the time necessary to fly over v_k and to process the images).

We now define two functions that exploit the concept of completion time of a site and are useful to evaluate the goodness of a solution: in the first one we generalize a classical parameter (*i.e.* latency) in order to take into account the priority, while the second one is the classical definition of a very natural optimization function, that is completion time.

Definition 5: The *weighted latency of a solution* SOL^e, $wL(\mathrm{SOL}^e)$, is the mean of the completion times of all sites (either in the first or in the second scenario), taking into account their priorities:

$$wL^{(i)}(\mathrm{SOL}^e) = \frac{1}{n}\sum_{v\in V} p(v)cost^{(i)}(v, \mathrm{SOL}^e),\ \ i = I, II. \quad (8)$$

Changing parameters $p_{min}, p_{med}, p_{max}$ influences how much it is pressing to serve higher priority sites first. Namely, sites with the highest priority should be served first, in order to associate a higher multiplicative factor (corresponding to the priority) to a lower sum of costs. Hence, we can claim the following:

Problem Requirement HPSF (Higher Priority Sites First): *The sites with higher priority should be flown over before than the ones with lower priority.*

Definition 6: We denote as the *completion time* of a solution SOL as:

$$ct^{(i)}(\mathrm{SOL}^e) = \max_{v\in V} cost^{(i)}(v, \mathrm{SOL}^e) \text{ where } i = I, II. \quad (9)$$

Clearly, we consider a solution better than another one if it has smaller values of both ct and wSOL. Observe that $wL^{(i)}(\mathrm{SOL}^e)$ and $ct^{(i)}(\mathrm{SOL}^e)$ are not independent indeed, calling $wA(p) = \frac{1}{n}\sum_{v\in V} p(v)$ the weighted average of the priorities over all nodes, it holds that $wL^{(i)}(\mathrm{SOL}^e) \leq wA(p)\cdot ct^{(i)}(\mathrm{SOL}^e)$.

Note that $ct^{(i)}$ does not take into account the priority of nodes; nevertheless, it is a primary parameter to evaluate the goodness of the solution.

We conclude this subsection with the formal definition of our problem:

Definition 7 (CMP): Given a complete node- and edge-weighted graph $G = (V, E, dist, \sigma, p)$, defined as in Subsection II-C, and a set of q homogeneous UAVs $u_1, \ldots, u_q$, the problem *Cover by Multitrips with Priorities (CMP)* consists of finding a set SOL^e of q effective sequences such that $ct^{(i)}(\mathrm{SOL}^e)$, where $i = I, II$, is minimized and Requirement HPSF is addressed.

It is easy to see that CMP is a generalization of the well known multiple Travelling Salesperson Problem, m-TSP [30] (achievable with $B = \infty$, $p(v) = 1$ and $\sigma'(v) = 0\ \forall v$) so, in its turn, it is strongly NP-hard. Nevertheless, the wide literature devoted to m-TSP and all its variants cannot be exploited in this case; indeed, this class of problems misses an important parameter, that is the battery constraint B. Modifying algorithms for m-TSP to address the battery constraint does not seem an easy issue as pointed out in [28] for a different problem, and it seems neither possible to inherit from m-TSP any approximability result without relaxing the battery constraint (and accepting, for instances, cycles with completion time $B+\epsilon$ for some $\epsilon > 0$), that in our context is inadmissible.

E. A MIXED-INTEGER LINEAR PROGRAMMING FORMULATION

We exploit the nomenclature defined in Subsection II-C to formulate CMP as a mixed-integer linear programming problem in the first scenario.

The following sets and parameters are known from the problem: (i) V: set of $n+1$ sites; (ii) Q: set of q UAVs; (iii) B: UAVs' battery capacity; (iv) $p(v_i)$: priority of site v_i.

Moreover, we define the following: (i) K: set of feasible (effective) cycles and $K' = K \cup \{k_0\}$ where k_0 is an empty cycle introduced for modeling reasons; for all $C_k \in K$, $t_k(v_i)$ (completion time for site v_i on cycle C_k) and $t(C_k)$ (flying time of cycle C_k) are considered as known; (ii) Cov_i: set of cycles passing through site i; (iii) ϵ: a very small constant; (iv) M: a very large constant.

We introduce the following decision variables: (i) $X_k \in \{0, 1\}$, $k = 1, \ldots, |K|$, is a binary variable assuming value equal to 1 if cycle C_k is selected in the solution and 0 otherwise; (ii) $Y_{kj} \in \{0, 1\}$, $k = 1, \ldots, |K|$ and $j = 1, \ldots, q$, is a binary variable assuming value equal to 1 if cycle C_k is executed by UAV j; (iii) $Z^j_{k_1k_2} \in \{0, 1\}$, $k_1, k_2 = 0, \ldots, |K|$ and $j = 1, \ldots, q$, is a binary variable assuming value equal to 1 if cycle C_{k_1} and cycle C_{k_2} are executed in sequence by UAV j and 0 otherwise; (iv) $t_s(C_k)$, $k = 0, \ldots, |K|$, is a non-negative variable representing starting time of cycle C_k; (v) τ is a non-negative variable representing the total completion time; (vi) $cost(v_i)$ is a non-negative variable representing latency of site i.

Note that $t_s(C_k)$ and $cost(v_i)$ are not independent variables, on which we can directly act, as they are expressed as a function of other decision variables. Nevertheless, in a mathematical programming model, they must be defined as decision variables as well. Indeed, in an integer programming model, two types of entities are considered: input data, which are fixed and cannot be modified by the model, and decision variables, whose value must be decided by the model. The latter group includes both independent variables (such as those related to the selection and assignment of cycles) and variables that are expressed as a function of them (such as the total cost or the starting time of a cycle).

We claim that, in the first scenario, CMP can be completely described by the following set of constraints:

$$\min \quad \tau + \epsilon \sum_{v_i\in V} cost(v_i)p(v_i) \quad \text{(of)}$$

$$\sum_{C_k\in Cov_i} X_k = 1 \quad \forall v_i \in V \quad \text{(C1)}$$

$$\sum_{j\in Q} Y_{kj} = X_k \quad \forall C_k \in K \tag{C2}$$

$$\sum_{k_1\in K'} Z^j_{k_1k_2} = Y_{k_2j} \quad \forall C_{k_2} \in K \;\; \forall j \in Q \tag{C3}$$

$$\sum_{k_1\in K'} Z^j_{k_1k_2} = \sum_{k_1\in K'} Z^j_{k_2k_1} \;\; \forall C_{k_2} \in K \;\; \forall j \in Q \tag{C4}$$

$$\sum_{C_k\in K'} Z^j_{0k} = \sum_{C_k\in K'} Z^j_{k0} = 1 \quad \forall j \in Q \tag{C5}$$

$$t_s(C_{k_2}) \geq t_s(C_{k_1}) + t(C_{k_1}) - M(1 - \sum_{j\in Q} Z^j_{k_1k_2})$$

$$\forall\, C_{k_1} \in K \;\; \forall\, C_{k_2} \in K' \tag{C6}$$

$$t_s(C_k) \geq 0 \quad \forall C_k \in K \tag{C7}$$

$$\tau \geq t_s(C_k) + t(C_k) - M(1 - X_k) \tag{C8}$$

$$cost(v_i) \geq t_s(C_k) + t_k(v_i) - M(1 - X_k)$$

$$\forall\, v_i \in V \;\; \forall\, C_k \in K \mid C_k \in Cov_i \tag{C9}$$

The hierarchic objective function is reported in Equation (of). This function firstly aims at minimizing the total completion time and secondly at minimizing targets' weighted latency. The very small constant ϵ is defined such that a solution with a lower total completion time would be always preferred w.r.t. one with a higher one, whichever is the value of the secondary objective. This is a standard technique commonly used in multi-objective optimization dealing with hierarchical objective functions [31]–[33].

Constraints (C1) guarantee that each target is covered by exactly one cycle while Constraints (C2) impose that, if a cycle is selected, it must be assigned to exactly one UAV. Similarly, a cycle must occupy a position in the schedule list for a given UAV j if and only if it has been assigned to that UAV (Constraints (C3)). Constraints (C4) delineate the sequence of cycles for each UAV. If variable Z^j_{0k} is equal to 1 it means that k is the first cycle executed by UAV j. Similarly, if variable Z^j_{k0} is equal to 1, k is the last cycle executed by j. Constraints (C5) imply that each UAV is used and explicates that, for each UAV, there is a cycle that is flown over as the first one and a cycle that is flown over as the last one. These two cycles can coincide in the case of single-cycle UAV routes.

Starting times of cycles are ruled by Constraints (C6) and (C7): a cycle can start only after the previous cycles assigned to the same UAV has been completely flown over. Note that the order relationship between the starting times of the different cycles k_1 and k_2 is given by the term $M(1 - \sum_{j\in Q} Z^j_{k_1k_2})$, that guarantees the correctness of this constraint. Indeed, if C_{k_1} and C_{k_2} are executed in sequence by the same UAV, then $Z^j_{k_1k_2} = 1$ and thus we have $t_s(C_{k_2}) \geq t_s(C_{k_1}) + t(C_{k_1})$, thus guaranteeing that cycle C_{k_2} can only starts after the end of the previous cycle. On the other hand, if C_{k_1} and C_{k_2} are either executed by different UAVs or are not consecutive, we have that $Z^j_{k_1k_2} = 0$ and thus $t_s(C_{k_2}) \geq t_s(C_{k_1}) + t(C_{k_1}) - M$, making this constraint not binding.

The total completion time is computed through Constraint (C8). Finally, Constraints (C9) allow one to identify the latency of each site.

Note that M is a big constant, large enough to make Constraints (C6) actually binding only if cycle k_2 is performed by UAV j just after cycle k_1. A similar role is played by M also in Constraints (C8), which are actually binding only if cycle k is performed, *i.e.* only if $X_k = 1$.

The model involves $|K|+|K|q+|K|^2q$ binary variables and $|K|+n+1$ continuous variables. The number of constraints is $n+|K|+2|K|q+q+|K|^2+n|K|+1$.

If we want to model the second scenario, we can run the following *recovery procedure*: we solve the model as if we were in the first scenario and send all the UAVs following the optimal solution one cycle at the time; whenever a UAV is not able to fly over all the sites assigned to it, we construct a reduced graph starting from the input one and removing all the sites effectively visited together with their incident edges; we solve again the model on this reduced graph and send again all the UAVs following the new solution.

Finally, the model can be adapted to the case of a heterogeneous fleet of UAVs, characterized by a different battery duration and consumption rate. Indeed, feasible trips can be generated considering the less binding type of UAV, *i.e.* the one performing the longest duration trips. Then a set of constraints can be added in order to allow us to assign a cycle j to a UAV k only if the battery of k is enough to completely fly over j. We can define the compatibility between cycle j and UAV k as an input constant ϕ_{kj}, equal to 1 if UAV k can perform cycle j and equal to 0 otherwise. Then, to avoid forbidden assignments, we add the following set of constraints to the model:

$$Y_{kj} \leq \phi_{kj} \quad \forall j \in J \;\; \forall k \in K. \tag{10}$$

III. A META-ALGORITHM FOR CMP

In this paper, we will propose several heuristics, each one based on a different consideration. Nevertheless, all of them are built upon on the same meta-algorithm that is run at the operations center; it consists of a number of iterations going on until all sites have been flown over; at each iteration, a presumed cycle for each UAV is produced as output.

Here we describe an outline of this meta-algorithm, through some facts on which it is based:

• At the beginning, v_0 has complete knowledge of the structure of graph G, including the distances between the sites and the presumed values $\sigma(v)$, but (only in the second scenario) not the effective values $\sigma^e(v)$.

• The solution is computed in any case at v_0, in order not to overburden UAVs with computation and intercommunication issues. This is possible because UAVs require to come back to v_0 to either recharge or substitute their battery and there they can get new information, *e.g.* a new cycle to perform.

• The meta-algorithm consists of several iterations to be executed until all sites have been flown over. At each iteration, a certain algorithm $\mathcal{A}$ is run; $\mathcal{A}$ computes a set of (presumed) cycles, one for each UAV, and for each such cycle C, $t(C) \leq B$. Changing $\mathcal{A}$, either in one or in all the iterations, produces a different heuristic, as we will detail later in Section VI.

• In the second scenario, each UAV is equipped with a device (*e.g.* a radio) to directly communicate with the base. Such a device is primarily exploited to call rescue teams when it realizes that there are survivors to be rescued. Moreover, as soon as a UAV decides to go back to v_0, for example, because it has not had enough battery to continue, it sends a message indicating its latter overflown site, and v_0 will immediately update the set of sites to be still flown over. In this way, v_0 has in real-time a precise knowledge about the (im)possibility to successfully complete the flown over of the assigned cycle by each UAV.

• In the first scenario, presumed and effective cycles coincide, so the iterations of the meta-algorithm can be run all together, outputting a final solution.

In the second scenario, presumed and effective cycles can be even very different, so the iterations are performed one by one, and the next iteration is run only after that UAVs inform the operations center v_0 of the effective cycles. Namely, at each iteration, a cycle for each UAV is computed at v_0 and it is assigned as a presumed cycle; it is assumed that all the sites in these assigned cycles will be flown over; therefore sites currently assigned to a UAV are considered committed and cannot be assigned to any other UAV. Based on the effective value of σ' on the already flown over sites in the current cycle and of the remaining battery, every UAV can periodically infer whether some sites assigned to it will remain not flown over. Specifically, before visiting the next node, a UAV is able to calculate if the remaining energy is not enough to continue the trip and thus it has to return to v_0. All the remaining nodes in the presumed cycle will be automatically discarded from the current effective cycle. The UAV communicates this information to v_0, where the not flown over sites will be released and included again in the current graph; then, a new set of presumed cycles is computed. In this way, as soon as each UAV comes back to the base, it will have ready a new presumed cycle to be assigned to it. This iteration continues until all the sites have been flown over.

In the following sections, we will show that this meta-algorithm can also be used as a preprocessing algorithm to make some preliminary estimates of the required completion time and the best number of additional batteries and UAVs required.

IV. CHANGING VS RECHARGING BATTERIES

Suppose that $b \geq 0$ additional batteries are available for the q UAVs. Assume that the meta-algorithm is run immediately after an earthquake as a pre-processing and a (presumed) solution SOL $= \{\mathcal{S}_{u_1}, \cdots, \mathcal{S}_{u_q}\}$ is output; we call $N(\text{SOL}) = \sum_{i=1}^{q} |\mathcal{S}_{u_i}|$ the total number of cycles in SOL. As SOL is clear from the context, we call it simply N.

In all this section we assume that $\max_{u_i} |\mathcal{S}_{u_i}| > 1$, otherwise there were no need of additional batteries, and $\mathcal{S}_{u_{ct}} = \{C_1(u_{ct}), \ldots, C_{y_{ct}}(u_{ct})\}$ is the sequence flown over by UAV u_{ct} whose cost equals the completion time of SOL, and its last site before coming back to v_0 is v_{ct}; in other words, $C_{y_{ct}}(u_{ct})$ is the last cycle to be oversight in the sequence assigned to u_{ct} and v_{ct} is the last site lying on it. (Note that sequence $\mathcal{S}_{u_{ct}}$ does not necessarily coincide with the one with the maximum number of cycles.)

We now analyze each possible value of b and associate to it the corresponding value of the completion time, so that civil defense organizers can have an immediate idea about the necessary time to conclude operations, knowing all the parameters. We derive also the minimum number of batteries b necessary to guarantee minimum completion time (*i.e.* no time in which UAVs remain inactive waiting for battery recharging).

We focus first on the first scenario and, for the sake of simplicity in calculations, we assume that every cycle of each sequence of every UAV is flown over in time exactly B. This is an upper bound on the completion time of cycles, but this approximation is, in fact, negligible especially for some heuristics, as shown in Table 1, so we use it to simplify our calculations. It is not difficult to see that more precise values give anyway similar results.

TABLE 1. Average times of the cycles of a solution computed by the different heuristics proposed in this paper. This table is computed with $q = 5$, $B = 50$, uniform distribution of the sites; the average is computed on 20 runs. (We refer to Section VI for the formal definition of $\mathcal{H}_{TSPN}$, $\mathcal{H}_{TOP}$, $\mathcal{H}_{Greedy}$, $\mathcal{H}_{mixed}$, $\mathcal{H}_{IP}$.)

n	$\mathcal{H}_{TSPN}$	$\mathcal{H}_{TOP}$	$\mathcal{H}_{Greedy}$	$\mathcal{H}_{mixed}$	$\mathcal{H}_{IP}$
50	37.10	43.67	48.71	45.66	46.66
75	38.88	44.58	48.82	47.41	46.75
100	40.26	44.71	48.82	47.54	47.57
125	41.11	45.35	49.00	47.67	48.11
150	41.78	44.97	48.89	47.93	48.38
175	42.40	45.44	48.99	48.13	48.51
200	42.37	45.48	48.97	48.31	48.53

We distinguish different cases according to the relation between b and q.

case $b = 0$: If there are no additional batteries, the completion time is trivially given by $\sum_{i=1}^{y_{ct}-1}(t(C_i)+R)+t(C_{y_{ct}})+t^{(I)}(v_{ct}) \leq (y_{ct}-1)(B+R)+2B$. Informally, we can read this result as follows: we consider time as marked in intervals $B+R$ long, and an immediate upper bound on the maximum time necessary to complete operations by solution SOL is given by the product between $(B+R)$ and y_{ct}; in view of the definition of completion time, a more precise approximation substitutes the last R long interval with $t^{(I)}(v_{ct})$ that is bounded by B. As a side effect, this calculation proves also the following finite-time result:

Theorem 1 (Termination 1st sc): In the first scenario, there always exists a solution to CMP and an upper bound to its completion time is given by $(y_{ct}-1)(B+R)+2B$ (reached in absence of additional batteries and when $t(C) = B$ for every C in SOL).

case $0 < b < \min\{\lceil \frac{R}{B} \rceil q, N-q\}$**:** If there are few additional batteries (where "few" means that they are not enough to cover all the operations without UAVs' inactivity time due to recharging batteries), b is necessarily less than:

- $N - q$, that is the total number of flights in SOL after the subtraction of the first one for each UAV, performed with the supplied battery;
- $\lceil \frac{R}{B} \rceil q$, because more batteries would imply no UAVs' inactivity due to recharge operations.

Observe that the value of $ct^{(I)}(\text{SOL})$ as function of b is not decreasing (in other words, for any $k > 0$, the completion time using $b + k$ additional batteries is not greater than the completion time using b additional batteries). We will use the monotonicity of this function to simplify calculations and consider only the cases when b is a multiple of q. In this special case, the whole fleet of UAVs will be able to go on and back exactly $\frac{b}{q} + 1$ times without waiting for recharging batteries. Every time the q UAVs come back to the base, exactly q batteries are put to be recharged. After $\frac{b}{q} + 1$ flights, all UAVs (and among them u_{ct}, so it is not restrictive to focus on it) necessarily have to wait for the first batteries to be completely recharged; in this case, the waiting time of u_{ct} due to battery recharge after cycle $C_{\frac{b}{q}+1}(u_{ct})$ represents the waiting time after the first group of flights and is $wtb(1) = R - \sum_{i=2}^{\frac{b}{q}+1} t(C_i(u_{ct}))$. After this time, u_{ct} (and analogously all the other UAVs) will be able to perform another group of $\frac{b}{q} + 1$ flights without any waiting time. More in general, the waiting time after the r-th group of flights is $wtb(r) = R - \sum_{i=(r-1)(\frac{b}{q}+1)+2}^{r(\frac{b}{q}+1)} t(C_i(u_{ct}))$.

The number of groups of flights that need to be followed by a waiting time are $\lceil \frac{y_{ct}}{\frac{b}{q}+1} \rceil - 1$, so the completion time is

$$ct^{(I)}(\text{SOL}) = \sum_{i=1}^{y_{ct}} t(C_i(u_{ct})) + \sum_{r=1}^{\lceil \frac{y_{ct}}{\frac{b}{q}+1} \rceil - 1} wtb(r) + t(v_{ct}) \qquad (11)$$

$$\leq \left(\left\lceil \frac{y_{ct} \cdot q}{b+q} \right\rceil - 1 \right)(R + B) + B(\frac{b}{q} + 2). \qquad (12)$$

case $b \geq \min\{\lceil \frac{R}{B} \rceil q, N - q\}$**:** It is not difficult to see that, with so many additive batteries, it is possible to let UAVs fly without pauses, because there are always enough charged batteries. In this case, the completion time is upper bounded by $ct^{(I)}(\text{SOL}) = \sum_{i=1}^{y_{ct}-1}(t(C_i)) + t(C_{y_{ct}}) + t^{(I)}(v_{ct}) \leq B(y_{ct} + 1)$.

This proves the following result:

Theorem 2 (Completion Time Minimization): In the first scenario, a guarantee for minimizing the completion time is to have at least $\lceil \frac{R}{B} \rceil \cdot q$ additional batteries and to minimize the value of y_{ct}.

Figure 2 depicts the comparison between the function describing completion time in the first scenario when b grows up from 0 to $\lceil \frac{R}{B} \rceil \cdot q$ on an example instance and the upper bounds provided above. The gap between the two plots in the

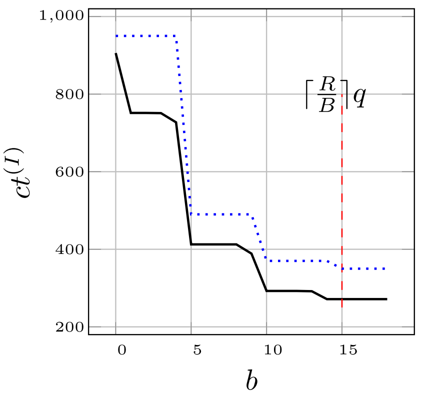


FIGURE 2. Plots of the function describing $ct^{(I)}$ (solid line) in minutes in an instance with $B = 50$, $R = 120$, $q = 5$, $n = 200$ whose solution has $N = 26$ and $y_{ct} = 6$ and of the found upper bounds (dotted line) in the first scenario.

interval between 0 and $\lceil \frac{R}{B} \rceil q$ is due to the approximation we did upper bounding $ct^{(I)}(\text{SOL})$ with its values reached when q divides b; this is also the reason why the corresponding function is a step function. Nevertheless, our computation is quite good when we need to estimate the number of needed additional batteries to avoid waiting times due to recharging operations.

All the previous considerations cannot be applied to the second scenario as they are, because the pre-processing phase outputs only presumed cycles. In the following, we estimate the average of an additional contribution that cannot be ignored when passing from a presumed to an effective solution.

Assume that, immediately after the earthquake, we are able to approximately quantify a uniform probability p to find people needing help during the flight and that, for simplicity, $\sigma(v)$ is a constant σ and $\sigma'(v)$ is either 0 or a constant σ', for every $v \in V$. Given a presumed solution SOL, with N presumed cycles output by a pre-processing computation, the average of the number of sites in each cycle is $E[m] = \frac{n}{N}$ and the $E[m] + 1$ sites lie on every cycle at an average distance of $\frac{B - E[m] \cdot \sigma}{E[m]+1}$. For every presumed cycle C of SOL, on average, in $E[m] \cdot p$ sites a UAV will detect a need to spend a further time σ' long.

When passing from C to the corresponding effective cycle C^e (containing $m' \leq m$ sites), $m - m'$ sites will not be in fact flown over, and the average value for m', $E[m']$, is the maximum value for which it holds:

$$E[m']\sigma + (E[m'] + 1)\frac{(B - E[m]\sigma)}{E[m] + 1} + pE[m]\sigma' \leq B. \qquad (13)$$

Executing some algebraic calculations, we get:

$$E[m'] \leq \frac{Bn + BN - p\frac{n^2}{N}\sigma' - pn\sigma' - BN + n\sigma}{n\sigma + \sigma N + BN - n\sigma}$$

$$\leq \frac{n}{N} - \frac{n}{N} \cdot \frac{p\sigma'(n + N)}{N(B + \sigma)}. \qquad (14)$$

In average, $E[m-m']\cdot N = (E[m]-E[m'])\cdot N$ sites remain not served and, in average, they will require a further number of cycles equal to $\frac{(E[m]-E[m'])N}{E[m']}$ that is lower bounded as follows (upper bounding $E[m']$ with $E[m]{=}n/N$):

$$\frac{(E[m]-E[m'])N}{E[m']} \geq \frac{p\sigma'(n+N)}{B+\sigma}. \tag{15}$$

Now, calling $N'(\mathrm{SOL}, p, \sigma, \sigma') = \frac{p\sigma'(n+N)}{B+\sigma}$, the total number of estimated cycles is at least $N + N'(\mathrm{SOL}, p, \sigma, \sigma')$. If we substitute this value instead of N in the computations related to the first scenario and modify them according to the definition of completion time of a site in the second scenario, we have an estimate on $ct^{(II)}(\mathrm{SOL})$ w.r.t. the number of batteries b. We explicitly write only the following, that is the analogous of Theorem 1 and provides also a finite-time result because Theorem 2 can be re-written identical for the second scenario:

Theorem 3 (Termination 2^{nd} sc): In the second scenario, there always exists a solution to CMP and an estimate to its completion time is given by $\lceil (N{+}N'(\mathrm{SOL}, p, \sigma, \sigma'))/q\rceil (B{+}R){-}R$ (reached in absence of additional batteries and when $t(C){=}B$ for every $C \in \mathrm{SOL}$).

In our experimental evaluation we observed that, although the derived formulas would provide lower bounds to the estimates of $ct^{(II)}(\mathrm{SOL})$, in practice they are quite close to the exact values, especially for small values of p, as shown in Table 2.

V. NUMBER OF UAVs TO GUARANTEE A SINGLE CYCLE

In this section we give a simple method to immediately estimate the number of UAVs that civil defense should own to ensure that all the operations can be concluded within a single UAV flight, *i.e.* without any changing or recharging batteries and within time B. To do this, we recall the definition of a well-known problem in the literature.

The *Team Orienteering Problem* (TOP) [34] comes from an outdoor sport played on mountains where a team of hikers, with the help of a map and a compass, must visit as many nodes as possible within a specified time limit, getting a profit from each visited node. Since the goal is to maximize the total gain that the hikers all together collect, if the gains associated with nodes are equal, equivalently the problem is to maximize the number of nodes visited. TOP is also known as *selective m-TSP* since not all nodes are visited in a solution.

More in detail, using our nomenclature, the *Team Orienteering Problem* (TOP) [34] can be defined as follows: let $G = (V, E, l, p)$ be a complete graph, where $V = \{v_0, v_1, \ldots, v_n\}$ is the set of nodes, v_0 is the base and the remaining nodes are objectives. Each node v_i, $i = 0, 1, \ldots, n$, is associated with a profit $p(v_i)$ ($p(v_0) = 0$), corresponding to our priority. There are q hikers and each one gains a profit $p(v_i)$ if visits a still unvisited v_i. A travel time $l(v_i, v_j)$ is associated to each edge (v_i, v_j), that is assumed to satisfy the triangular inequality. The hikers must complete their tour within a predetermined time B. It follows that some nodes may not be visited due to the B constraint, so TOP consists in determining a set of q cycles, each passing through v_0 and respecting the time constraint such that each node is visited at most once and the total profit collected is maximized. If the profit of all sites is the same, TOP maximizes the number of visited sites.

It is evident that, if after solving TOP on our instance, all the sites are visited, we are guaranteed that a single flight (*i.e.* a time-bounded by B) will be enough to complete all the rescue operations. We want to give an estimate on the number q of UAVs needed to make this possible.

Let us focus on the first scenario, and first put forward the hypothesis to have n UAVs. Because of the feasibility of the problem, $l(v_0, v) + l(v, v_0) \leq B$, so we are guaranteed that each UAV will fly over a single site, concluding the whole operations with a single flight and within time B. Now, we can insert the computation of an algorithm solving TOP on our instance inside a loop that mimics a binary search on the value of q; so, we assume now that $q = n/2$, run the algorithm and output a solution for TOP; if all the sites are visited by this solution, then we eliminate all the values larger than current q, otherwise, we eliminate all the values smaller than current q, and then we will proceed again recursively with a new q equal to the median of the remaining values. This algorithm, together with the fact that presumed and effective cycles coincide in the first scenario, guarantees that, in a number of iterations logarithmic in n, we have the exact value of q allowing us to get a solution in which all UAVs fly for a single cycle.

Concerning the second scenario, we have no knowledge in advance about the exact number of cycles in an effective solution, so we can only estimate this number similarly to how we did in the previous section adding the contribution of $N'(\mathrm{SOL}, p, \sigma, \sigma')$ and mime again a binary search. In this case, we will get an expected value of q.

We formalize the result of this section as follows:

Theorem 4 (Number of UAVs for a Completion Time Bounded by B): By running an algorithm solving TOP a number of times upper bounded by $\log n$, it is possible: (i) in the first scenario, to exactly determine; (ii) in the second scenario, to have an estimate on the number of UAVs needed to ensure the end of the rescue operations within time B.

Unfortunately, TOP is *NP*-hard and *APX*-hard [35]. So, there is a wide literature solving the problem either optimally or with good approximations (see *e.g.* the survey [36]). Nevertheless, even these latter algorithms require very long times, though theoretically polynomial. Thus, we use a very simple and fast heuristic for TOP [37], which guarantees an immediate calculation of the cycles, at cost of a slightly worse solution; indeed the obtained results of this heuristic have a similar quality as the results of the best-known heuristics while the computational time is reduced significantly. Of course, the better the guaranteed approximation of the algorithm used to solve TOP, the more accurate the number of UAVs produced by the algorithm described above will be.

TABLE 2. Average presumed (left), real (middle) and estimated (right) total number of cycles computed by $\mathcal{H}_{TSPN}$, $\mathcal{H}_{TOP}$, $\mathcal{H}_{Greedy}$ and $\mathcal{H}_{mixed}$, for $B = 50$, $p = 0.25$, uniform distribution of nodes in the second scenario; the average is computed on 20 runs with $\sigma, \sigma' \in (0, 3]$. (We refer to Section VI for the formal definition of $\mathcal{H}_{TSPN}$, $\mathcal{H}_{TOP}$, $\mathcal{H}_{Greedy}$ and $\mathcal{H}_{mixed}$.)

	$\mathcal{H}_{TSPN}$			$\mathcal{H}_{TOP}$			$\mathcal{H}_{Greedy}$			$\mathcal{H}_{mixed}$		
n	N	real	estim.	N	real	estim.	N	real	estim.	N	real	estim.
50	22.15	23.05	22.16	15.65	16.90	15.66	10.35	12.75	10.36	12.20	14.15	12.21
75	28.30	29.95	28.32	21.00	22.45	21.02	14.90	17.25	14.91	15.30	18.30	15.31
100	34.10	36.25	34.12	27.30	28.45	27.32	18.25	21.25	18.27	20.05	23.20	20.07
125	39.25	42.85	39.28	32.30	34.20	32.32	20.25	25.55	20.27	23.30	26.75	23.32
150	44.35	48.65	44.38	38.15	39.75	38.18	24.45	30.15	24.48	26.15	31.00	26.18
175	49.45	55.05	49.49	43.20	45.30	43.23	27.10	33.45	27.13	29.95	35.05	29.98
200	55.00	60.85	55.04	48.30	50.85	48.34	30.50	37.35	30.54	33.25	38.90	33.29

VI. HEURISTICS

In this section, we present five heuristics to solve our problem both in the first and in the second scenario. Every time each of them is executed, it produces as output q node-disjoint cycles (except v_0, from which all of them pass), $C(u_1), \dots C(u_q)$ with the properties that $t(C(u_i)) \leq B$ and Problem Requirement HPSF is somehow addressed.

Once more we observe that the meta-algorithm presented in Section III produces a feasible and definitive solution in the first scenario, after which the algorithm $\mathcal{A}$ is defined. As for the second scenario, the meta-algorithm is run in iterations so that the cycles of the next iteration are computed only after that it is known which portion of the cycles of the previous iteration has been in fact flown over and the input graph of $\mathcal{A}$ at each iteration is the subgraph induced by the sites that are still uncovered.

A. TOP BASED ALGORITHM $\mathcal{A}_{TOP}$

Before describing algorithm $\mathcal{A}_{TOP}$, we present a case study from which it is possible to derive a general rule that we would like to follow to get better solutions.

Assume to have an instance (either of the first or of the second scenario) in which many nodes $v_1, v_2, \dots v_{n-1}$ lie very close to v_0, while a single node v_n is far from it; moreover, due to the distances between v_0 and all the other nodes, it is possible to group the sites in only two cycles: $C_1 = \langle v_0, v_1, v_2, \dots, v_{n-1}, v_0\rangle$ and $C_2 = \langle v_0, v_n, v_0\rangle$; of course, it holds that $t(C_1) \leq B$ and $t(C_2) \leq B$. For simplicity, we assume that $q = 1$, that the n nodes have all the same priority (w.l.o.g. equal to 1) and that $\sigma(v_i) = \sigma^e(v_i)$ for each $i = 1, \dots, n$. There are only two possibilities for the solutions: C_1 is executed either as first (SOL$_1$) or as a second cycle (SOL$_2$). It is easy to see that $ct(\text{SOL}_1) = ct(\text{SOL}_2)$ while $wL(\text{SOL}_1)$ is much lower than $wL(\text{SOL}_2)$, indeed in SOL_2, the time contribution of flying over the first cycle followed by the possible recharge time turns out to be multiplied for the number of sites flown over in the successive cycle. More precisely, in the first scenario: $n \cdot wL^{(I)}(\text{SOL}_1) = n \cdot t(C_1) + R + t(C_2) + \sum_{i=1}^{n} t(v_i)$ and $n \cdot wL^{(I)}(\text{SOL}_2) = n \cdot t(C_2) + (n-1)(R + t(C_1)) + \sum_{i=1}^{n} t(v_i)$ while in the second scenario: $n \cdot wL^{(II)}(\text{SOL}_1) = \sum_{i=1}^{n}(t(v_i) + \sigma'(v_i)) + t(C_1^e)$ and $n \cdot wL^{(II)}(\text{SOL}_2) = \sum_{i=1}^{n}(t(v_i) + \sigma'(v_i)) + (n-1)(t(C_2^e) + R)$.

This is a very special example with many simplifications, but the result can be generalized as follows: *The cycles with a greater number of sites should be ante-posed to the other cycles in a solution.*

We observe that TOP (defined in Section V) aims at maximizing the profit, which results either in choosing high priority sites or in choosing a larger number of low priority sites; both cases go in favor of the minimization of wL.

We select as $\mathcal{A}_{TOP}$ the heuristic described in [37], having time complexity $\mathcal{O}(|E| \log |V| + q|V|^2) = \mathcal{O}(n^2(q + \log n))$: indeed, to the best of our knowledge, every approximation algorithm for TOP has a very high computational time in practice and so cannot be used; instead the chosen heuristic turns out to have a similar quality as the results of the best-known approximation algorithms while the computational time is reduced significantly.

B. GREEDY BASED ALGORITHM $\mathcal{A}_{Greedy}$

This algorithm finds all the q cycles at the same time according to a greedy approach. Namely, it first fixes an *a priori* order on the UAVs, then, at each step it considers one by one all the q current cycles and, for each of them, starting from the site selected last v_{last} (at the beginning v_0), selects the next site v_{next} as the one maximizing the quotient $p(v_{next})/l(v_{last}, v_{next})$ still guaranteeing that the whole cycle with the addition of v_{next} can be flown over within time B.

The reason why we decided to maximize $p(v_{next})/l(v_{last}, v_{next})$ is that we would like to choose sites with high priority (hence addressing rule HPSF) without going too far, whenever possible, to keep low the necessary number of cycles.

Calling s the number of sites involved overall in the q computed cycles, the computational complexity of this heuristic is $\Theta(ns) = \mathcal{O}(n^2)$.

C. IP FORMULATION BASED ALGORITHM $\mathcal{A}_{IP}$

We observe that, although elegant, neither for very small values of the number of sites the model described in Subsection II-E can be used in practice, because $|K|$ is exponential in the number of sites, and the number of Constraints (C3) and (C4) is quadratic in $|K|$. This is the reason why we propose a heuristic based on a modification of this model.

Namely, we divide the problem into two consecutive subproblems: first, we compute an optimal *cycle cover* of all the sites (where all cycles pass through v_0): the optimization function minimizes the number of cycles in the solution; secondly, we perform an optimal *scheduling* of the only cycles in this solution; in this case, we minimize the completion time.

Clearly, the solution to this variation is possibly worse than the optimal one. Since computing an optimal cycle cover is still a time-consuming issue, to speed up the computational time, even by deteriorating the quality of the solution, we do not generate the whole set of possible cycles K but a subset of it. Namely, when we are generating a cycle, we impose that every site can be followed in the cycle by one among the k sites closest to it, where k is a parameter to be manually tuned. Clearly, the largest is k, the larger will be the number of generated cycles, the closer the solution will be to the optimal one, but the larger will be the required computational time. To reach a good balance between efficiency and effectiveness, in our experiments we choose $k = 5$.

The algorithm consisting first of (not optimally) solving the cycle cover problem and then in (optimally) solving the scheduling problem on the solution of the first sub-problem represents algorithm $\mathcal{A}_{IP}$.

Solving the cycle cover phase in this simplified model corresponds to the minimization of the cumulative length of the selected cycles, subject uniquely to Constraints (C1). This model is much simpler to solve with respect to the complete model since it involves only $|K|$ binary variables, no continuous variables, and n constraints. In the scheduling phase, we apply the model proposed in Section II-E but on a restricted set of cycles, $K^\star$ containing only those selected by the simplified model adopted in the first phase. The cardinality of $K^\star$ can be several orders of magnitude lower than the cardinality of K making this subproblem much faster to be solved with respect to the original one.

D. ALGORITHM $\mathcal{A}_{TSPN}$

As discussed in the Introduction, no prior works are dealing with the scenarios we consider in this article. So, in order to compare the performance of our heuristics, we modified one of the best-performing algorithms in the literature, that is, one of the algorithms described in [4], [15]. In their work, Kim *et al.* designed many algorithms for solving different problems, all related to ours, but no one of them is the same. Among them, we selected the closest to ours, that is what they call q-TSPN (q-Travelling Salesman Problem with Neighborhood). The algorithm designed by the authors for this problem is one of the best solutions currently available. Indeed, unlike several previous proposals, it is not a heuristic but it is an approximation algorithm with a provable constant approximation ratio. Nevertheless, the considered problem requires weaker assumptions than ours, *i.e.* no battery constraints and no priorities for sites, so we need to modify their algorithm to compare it with our heuristics.

Note that the modifications we made to the original algorithm do not compromise its performance in any aspect, but

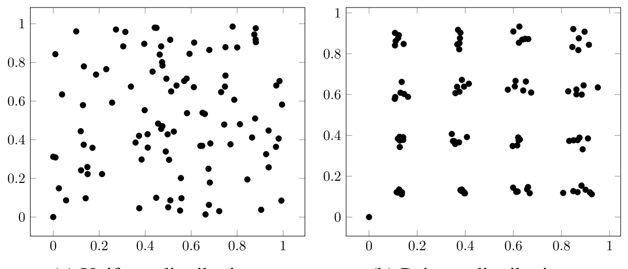


FIGURE 3. Two examples of instances in which $n = 100$ sites are positioned on a squared area, according to: (a) uniform and (b) modified Poisson distribution.

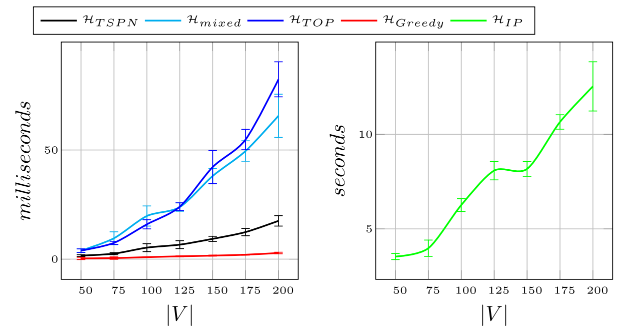
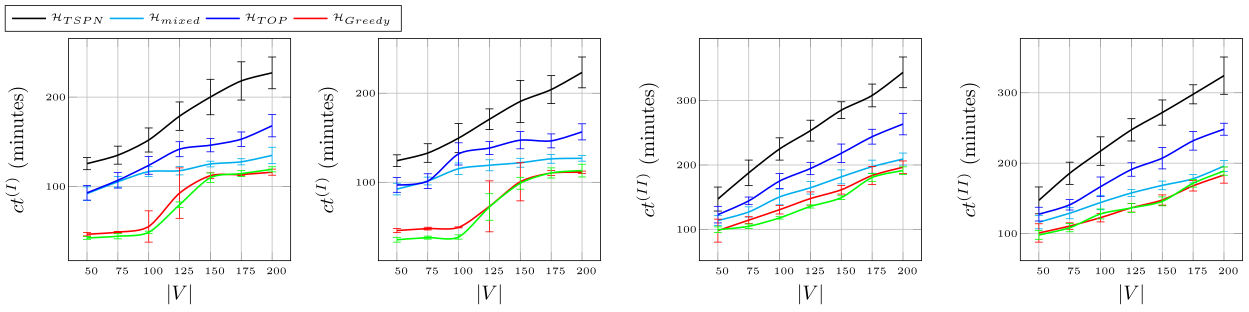


FIGURE 4. Execution time (average per Meta-algorithm's iteration) in the second scenario as a function of the number of sites; $q = 10$, $B = 50$, $p = 0.25$ and sites are uniformly distributed.

are only meant to extend its applicability to our two scenarios: indeed, running our modified algorithm with a very high value of B and with all equal priorities produces exactly the same output as the original algorithm. The original algorithm gives as part of the input q sites and uses them as children of the root of a minimum spanning tree rooted at v_0; the q sub-trees of this minimum spanning tree are then transformed into q cycles covering all sites and intersecting only in the root using the Christofides's approximation algorithm for TSP [38]; finally, some operations are executed in order to equalize the weight. We do not have in input the q sites, so we select them as a solution of *metric q-center problem* [39]: given set V with n sites and distance function l, the goal is to find a subset $C \subseteq V$ with $|C| = q$ such that the maximum distance of any point in V to the closest point in C is minimized. Unfortunately, the problem is NP-hard, and the algorithm we use guarantees an approximation ratio of 2 with a time complexity of $O(nq)$.

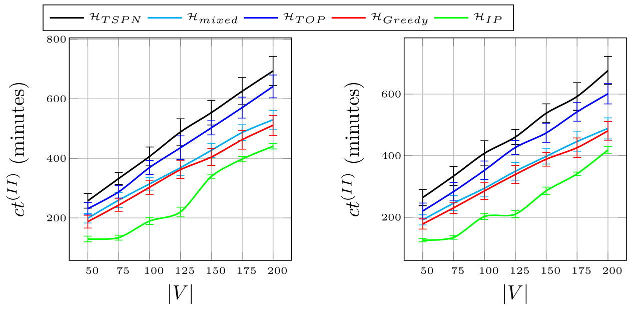
Moreover, to force this algorithm to take into account the site priorities, instead of working on the entire input graph, we consider it as the union of subgraphs, each one induced by the nodes with the same priority plus v_0: $G = G_{min} \cup G_{med} \cup G_{max}$, where G_i is the subgraph induced by $\{v_0\} \cup \{v \in V \mid p(v) = p_i\}$ and $i = min, med, max$. The aim is to choose first as many nodes with high priority as possible and to choose nodes with lower priority only later while considering the consumed budget.

Recalling that G is the graph induced by the set of sites that are still uncovered, this can be done by selecting the nodes



(a) uniform (b) Poisson (c) uniform (d) Poisson

FIGURE 5. Average completion time as a function of the number of sites; with parameters $q = 20, B = 50$ for the first scenario, and $q = 10, B = 50, p = 0.25, \sigma' \in (0, 3]$ for the second scenario.



(a) uniform (b) Poisson

FIGURE 6. Average completion time in the second scenario as a function of the number of sites; $q = 4, B = 60, \sigma' \in (7, 10], p = 0.25$.

to be added to the sub-trees from G_{max} first; after choosing as many nodes as possible in this graph, if the computed sub-trees have not been completed yet, the nodes will be selected from G_{med} and finally from G_{min}; of course, after that, the meta-algorithm has executed some times $\mathcal{A}_{TSPN}$, G_{max} will remain empty, hence, only nodes from the other graphs will be selected. When passing from the sub-trees to the cycles, the priorities are again kept into consideration, and the sites with high priority are placed along with the cycles before the sites with a medium priority which, in turn, are placed before the sites with low priority. This is necessary especially in the second scenario, in which there is no guarantee that all the sites in a cycle will be served in that cycle and, in this way, sites with lower priorities are possibly postponed to the next cycle. We call this algorithm $\mathcal{A}_{TSPN}$.

Observing that the update of the considered subgraph does not affect the worst-case time complexity (for which the dimension of the input is given by $\mathcal{O}(n)$ nodes and $\mathcal{O}(n^2)$ edges), the total cost of this algorithm is $\mathcal{O}(nq)$ to find the centers, plus the cost of finding the approximate TSP, $\mathcal{O}(n^2 \log n)$.

Before concluding this subsection, it is worth noting that, although the resulting algorithm is strongly inspired by [4], [15], we cannot guarantee anymore that the computed spanning tree is minimum; the reason is that we run this algorithm first on G_{max}, then on G_{med} and finally on G_{min}. It follows that its nice approximation ratio cannot be inherited in our scenarios.

E. HEURISTICS

We exploit algorithms $\mathcal{A}_{TSPN}$, $\mathcal{A}_{TOP}$, $\mathcal{A}_{Greedy}$ and $\mathcal{A}_{IP}$ to design five different heuristics, called $\mathcal{H}_{TSPN}$, $\mathcal{H}_{TOP}$, $\mathcal{H}_{Greedy}$, $\mathcal{H}_{mixed}$ and $\mathcal{H}_{IP}$: consider the meta-algorithm described in Section III; if $\mathcal{A}$ is equal to $\mathcal{A}_{TSPN}$ at each iteration, we get $\mathcal{H}_{TSPN}$; if $\mathcal{A}$ is equal to $\mathcal{A}_{TOP}$ at each iteration, we get $\mathcal{H}_{TOP}$ and if $\mathcal{A}$ is equal to $\mathcal{A}_{Greedy}$ at each iteration, we get $\mathcal{H}_{Greedy}$. Then, observe that TOP is not designed to be iterated, and so it gives its best during the first iteration, choosing first sites that are as close as possible to v_0 to maximize their number; this means that, while in the first iteration we select very attractive cycles, along with the successive iterations the solution of TOP could degrade in terms of goodness. This is the reason why we introduce a fourth heuristic, $\mathcal{H}_{mixed}$, running $\mathcal{A}_{TOP}$ in the first iteration of the meta-algorithm and $\mathcal{A}_{Greedy}$ in the successive ones.

Finally, in the first scenario, $\mathcal{H}_{IP}$ coincides with $\mathcal{A}_{IP}$ and we do not need to exploit the meta-algorithm. Vice-versa, in the second scenario, $\mathcal{H}_{IP}$ is obtained by running the meta-algorithm with $\mathcal{A}_{IP}$ if not at each iteration, at least each time we find a cycle for which presumed and effective versions do not coincide.

The difference of this algorithm w.r.t. the other ones is that, in this case, at each iteration of the meta-algorithm, a whole solution is output. This seems to be unavoidable for an approach based on the mixed-integer linear programming model and further contributes to increasing the computational time of this heuristic.

In Figure 1, a list of symbols used along the paper is reported. In the next section, we will show the result of several experiments geared to compare the performance of these five heuristics.

VII. EXPERIMENTS

All our experiments have been performed on a computer equipped with an Intel(R) Core(TM) i5-7200U CPU (4 cores clocked at 2.5GHz) and 8GB RAM; our programs have been implemented in C++ (g++ compiler v9.2.1 with optimization level O3).

We evaluate the performance of the proposed heuristics on two types of randomly generated graphs: we position $n \in [50, 200]$ points, placed: (i) either uniformly at random;

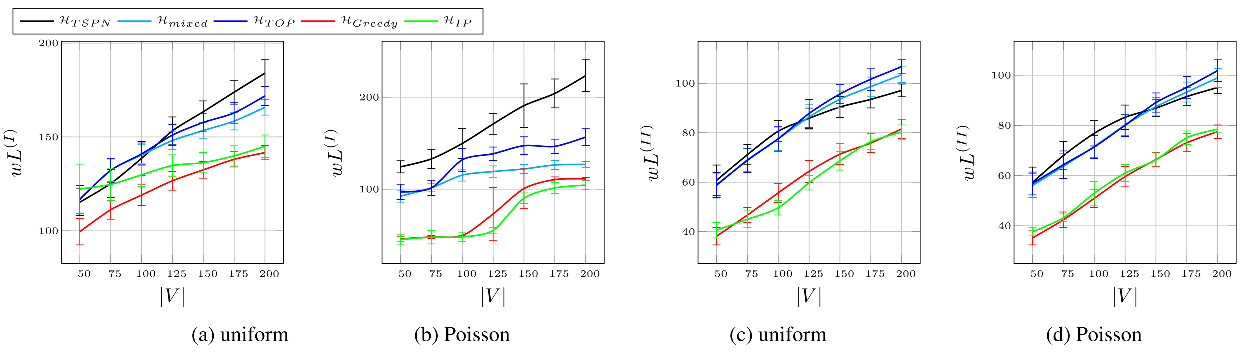


FIGURE 7. Average weighted latency as a function of the number of sites; with parameters $q = 20, B = 50$ for the first scenario, and $q = 10, B = 50, p = 0.25, \sigma' \in (0, 3]$ for the second scenario.

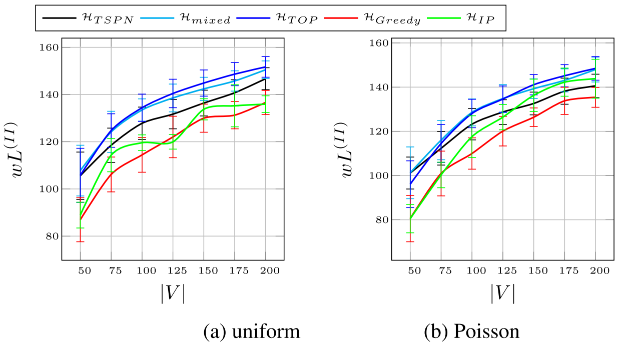


FIGURE 8. Average weighted latency in the second scenario as a function of the number of sites; $q = 4, B = 60, p = 0.25, \sigma' \in (7, 10]$.

(ii) or according to a Poisson point distributed method in a unit square modified as follows: we divide the unit square into 4×4 sub-areas and position the points in each sub-area using a Poisson point process. This distribution is intended to simulate building collapses concentrated in some specific zones.

Figure 3 depicts a graphical representation of $n = 100$ sites placed in a squared area when the two probability distributions are applied.

In all the experiments, we use the following parameter setting: $B = 50$ minutes (in agreement with the current technology, see *e.g.* [16]) and –to ease the reading of the results– the speed is chosen as 1000 meters per minute, corresponding to about 17 meters per second (in line with other recent papers in the literature, such as [40], [41]); $\sigma(v)$ is a real value randomly set (with uniform distribution) in the interval $(0, 3]$; when simulating the second scenario, we assign to sites a not null value of σ' randomly chosen with uniform distribution either in the interval $(0, 3]$ or in the interval $(7, 10]$ with probability p, p varying among three possible values: 0.25, 0.5 and 0.75. Only when $\sigma' \in (7, 10]$, we need to set $B = 60$, instead of 50, to guarantee that there exists a feasible solution. We choose a squared area of interest with dimension $15km \times 15km$, which always guarantees the feasibility of the problem, positioning v_0 at its bottom-left corner. Finally, we consider different values of q, coherently with the two scenarios: in the first one, we set $q = 5, 10, 15, 20$ while in the second one $q = 2, 4, 5, 10$.

In all the experiments dealing with the second scenario, the results are very similar with different values of p and q so we show the plots of only one setting, namely the one with $q = 10$ and $p = 0.25$.

All simulation runs have been repeated 20 times with different seeds, the plots show mean values with 95% confidence intervals.

First, we studied the running time of all the heuristics; as shown in Fig. 4 they do not behave all in the same way (in particular, we had to plot the running time of $\mathcal{H}_{\mathcal{IP}}$ separately for the sake of clarity because it is significantly higher than the other ones); nevertheless, the times are all low enough to consider all the heuristics equally suitable to be run in an emergency situation such as the one proposed in this paper. We remark that an approximated algorithm for TOP instead of $\mathcal{A}_{\mathcal{TOP}}$ would have required a time several orders of magnitude larger, and this would have been realistically inapplicable.

Then we studied the two performance metrics completion time and weighted latency.

As shown in Fig. 5 and 6, for all the heuristics the completion time grows almost linearly with the number of sites, but the growth rates of $\mathcal{H}_{IP}$, $\mathcal{H}_{Greedy}$ and $\mathcal{H}_{mixed}$ are lower than the other ones. Note that changing the values of parameters q and σ' 6 does not affect the trend of the completion times. The behavior similar to a step function of the completion time in some cases can be perhaps explained with the generation method of the graph instances (any graph with n nodes is constructed by an $(n - 1)$ node instance and so contains it as a sub-graph).

On the contrary, the weighted latency, see Fig. 7, and 8, does not always show a linear growth, due to its definition and to Problem Requirement HPSF, for which certain algorithms keep sites with high priority in the first cycles. Also for this function, $\mathcal{H}_{IP}$ and $\mathcal{H}_{Greedy}$ behave better than the other heuristics, and also, in this case, there is no valuable difference changing the value of q and of σ'. Although, note that when increasing the value of σ', $\mathcal{H}_{IP}$ tends to behave slightly worse w.r.t. $\mathcal{H}_{Greedy}$, especially when using the poisson distribution.

Finally, to highlight which heuristics best address HPSF, and to understand the different behavior of the heuristics, we considered the time in which all sites with a certain

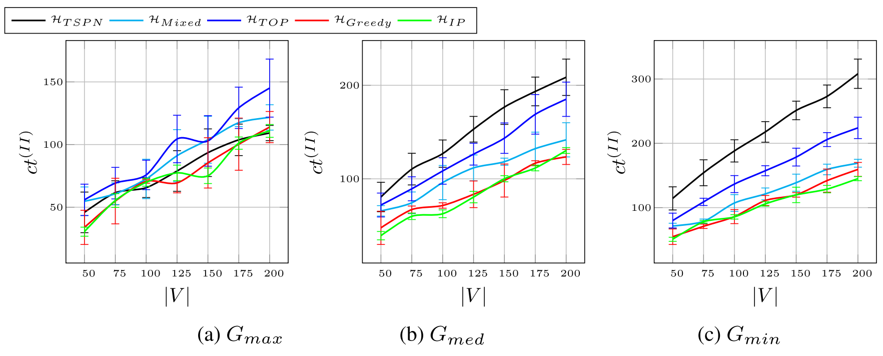


FIGURE 9. **Average completion time in the second scenario on the subgraphs G_{max} and G_{min} originated w.r.t. the priority of the sites. The simulation considered: $q = 10, B = 50, p = 0.25$ with sites uniformly distributed.**

priority have been served (see Fig. 9). Observe that $\mathcal{H}_{TSPN}$ behaves well in completely flying over the nodes of G_{max} but is then the last one to complete the operations (*i.e.* to completely fly over G_{min}); this is due to the adjustment of the algorithm from [4], [15] which forces it to process the high priority nodes first. $\mathcal{H}_{IP}$ and $\mathcal{H}_{Greedy}$ have the lowest completion times on all three subgraphs because both heuristics explicitly require site priorities to be taken into account. Note that the completion time of G_{min} does not coincide with the total completion time, since the time required to return to the base is not considered in Fig. 9(c).

In general, $\mathcal{H}_{TOP}$ is one of the least performing heuristics, for two different reasons: first, we recall that we are using the solution output by a heuristic since algorithms guaranteeing provably good solutions are too slow and cannot be used; second, TOP aims at maximizing the profit, and no importance is given to cycle equalization. This is, instead, an issue addressed by $\mathcal{H}_{Greedy}$, whose philosophy consists of filling up q cycles together. This is the reason why this algorithm performs better than all the others, besides being very simple and particularly fast.

We conclude this section by highlighting that $\mathcal{H}_{IP}$ heuristic performs better than the others from the point of view of performance metrics even if it is not far from $\mathcal{H}_{Greedy}$. On the other hand, $\mathcal{H}_{IP}$ is by far the slowest heuristic (3 orders of magnitude more than the others, which have similar behavior). So, we conclude that $\mathcal{H}_{Greedy}$ is the right compromise that balances all the metrics considered: its simplicity, low execution time, and good performance make it the best heuristic among those considered.

VIII. CONCLUSION AND OPEN PROBLEMS

In this paper we introduced a new problem, called Cover by Multitrips with Priorities, trying to model most of the parameters that emerge from the real-life situation of detecting people needing help immediately after an earthquake thanks to a fleet of UAVs.

We studied the problem theoretically, providing *a priori* estimates of the number of additional batteries to have waitings due to recharging batteries and the number of UAVs

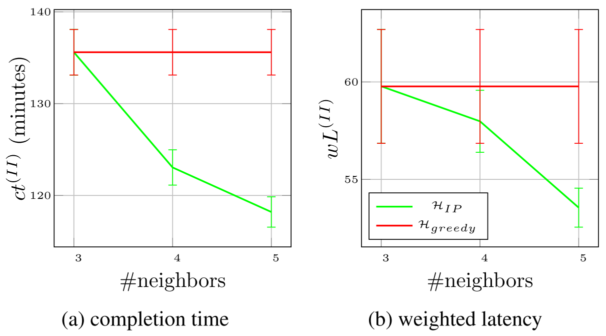


FIGURE 10. **Average completion time and weighted latency in the second scenario as a function of the number neighbors used to generate cycles in $\mathcal{H}_{IP}$, obtained using $n = 125, q = 10, B = 50, p = 0.25$.**

needed to have the time to complete all rescue operations delimited by B.

Then, we have proposed some heuristics on which we have carried out extensive experimentation, comparing them with each other and with one of the best-known algorithms that solve a similar problem; the results show that our heuristics significantly outperform this algorithm, although we added to it some parts to let it work well in our settings.

Many issues and possible future improvements naturally arise from this paper, and we already started to tackle some of them. Here is a non-exhaustive list:

- We considered a fleet of homogeneous UAVs; if they had different batteries and different cruise speeds and endurance, an easy approach would be to substitute in every computation the value of B with either $\min_i B_i$ (but we would get a worse performance of all the heuristics) or $\max_i B_i$ (but not all UAVs could terminate every cycle). An interesting open problem consists in modifying all the heuristics in order to let them work efficiently in the case of heterogeneous UAVs.
- We considered static distances; instead, distances could change in time in order to keep into account the weather and wind conditions. In this case, we can consider the graph with dynamic edge weights.
- As stated in Subsection VI-C, when we run heuristic $\mathcal{H}_{IP}$, we allow each site to be followed by $k = 5$ other sites and so generate cycles. We wondered how much the

goodness of the results is affected by this parameter. To this aim, we performed the experiments depicted in Fig. 10, showing that increasing the value of k significantly improves the performance metrics. Nevertheless, we expect that, for a certain value of k on, there will no longer be an appreciable improvement in performance metrics, that these values are very tight upper bounds for optimal values, and that we could hence consider them as benchmarks for testing heuristics and approximation algorithms.

Unfortunately, these limit values are not easy to achieve because both the execution time and the required RAM grow exponentially with k and this makes it unfeasible for any experiment with larger values of k.

• When we compute cycles of a solution we go on until there are sites that can be added and that guarantee to remain within the battery constraint B; so a UAV can return to base with a not-negligible amount of remaining battery that goes lost. We wondered if it was possible to accept that a site is partially flown over during a certain cycle to completely drain the battery of the corresponding UAV and then its exploration is completed at a later time (possibly by a different UAV). We decided not to implement this possibility since we have some case studies showing that this approach can be either favorable or not, depending on the relation between the main parameters of the problem. We are trying to give conditions on the parameters to delimit when partial overflights are convenient.

• From a methodological point of view, future research could address the design and development of other effective heuristic methods, such as Greedy Randomized Adaptive Search (GRASP) [42] or Multistart Random Constructive Heuristic (MRCH) [43] to generate potentially good cycles to be exploited by the IP model in heuristic $\mathcal{H}_{IP}$.

• Complete exploitation of UAVs' potentialities requires to introduce cooperation, both among them and between people and UAVs; these interactions would introduce new possibilities of producing faster and more flexible solutions.

ACKNOWLEDGMENT

The authors would like to thank Novella Bartolini, for the very interesting discussions concerning UAVs and their equipment, and Danilo Avola, updating them on image detection/recognition tools.

REFERENCES

[1] UAV Coach. *Master List of Drone Laws (Organized by State Country) L UAV Coach*. Accessed: Jan. 7, 2021. [Online]. Available: https://uavcoach.com/drone-laws/

[2] *Drones for Disaster Response and Relief Operations*, AR Cross, Wynne, China, 2012.

[3] C. Baker, S. Ramchurn, and W. Teacy, "Planning search and rescue missions for UAV teams," in *Proc. 22nd Eur. Conf. Artif. Intell.*, 2016, pp. 1777–1782.

[4] D. Kim, L. Xue, D. Li, Y. Zhu, W. Wang, and A. O. Tokuta, "On theoretical trajectory planning of multiple drones to minimize latency in search-and-reconnaissance operations," *IEEE Trans. Mobile Comput.*, vol. 16, no. 11, pp. 3156–3166, Nov. 2017.

[5] R. D. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence," *J. Int. Hum. Action*, vol. 3, no. 1, pp. 1–18, Dec. 2018.

[6] A. Albanese, V. Sciancalepore, and X. Costa-Pérez, "SARDO: An automated search-and-rescue drone-based solution for victims localization," *CoRR*, vol. abs/2003.05819, pp. 1–12, Mar. 2020.

[7] R. Avanzato and F. Beritelli, "A smart UAV-femtocell data sensing system for post-earthquake localization of people," *IEEE Access*, vol. 8, pp. 30262–30270, 2020.

[8] M. B. Bejiga, A. Zeggada, A. Nouffidj, and F. Melgani, "A convolutional neural network approach for assisting avalanche search and rescue operations with UAV imagery," *Remote Sens.*, vol. 9, no. 2, p. 100, Jan. 2017.

[9] J. Sun, B. Li, Y. Jiang, and C.-Y. Wen, "A camera-based target detection and positioning UAV system for search and rescue (SAR) purposes," *Sensors*, vol. 16, no. 11, p. 1778, 2016.

[10] E. T. Alotaibi, S. S. Alqefari, and A. Koubaa, "LSAR: Multi-UAV collaboration for search and rescue missions," *IEEE Access*, vol. 7, pp. 55817–55832, 2019.

[11] C. Wu, B. Ju, Y. Wu, X. Lin, N. Xiong, and G. X. Xu Liang, "UAV autonomous target search based on deep reinforcement learning in complex disaster scene," *IEEE Access*, vol. 7, pp. 117227–117245, 2019.

[12] G. Bevacqua, J. Cacace, A. Finzi, and V. Lippiello, "Mixed-initiative planning and execution for multiple drones in search and rescue missions," in *Proc. 25th Int. Conf. Automated Planning Scheduling*, 2015, pp. 315–323.

[13] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "UB-ANC planner: Energy efficient coverage path planning with multiple drones," in *Proc. Int. Conf. Robot. Autom.*, 2017, pp. 6182–6189.

[14] F. A. D. A. Andrade, A. R. Hovenburg, L. N. D. Lima, C. D. Rodin, T. A. Johansen, R. Storvold, C. A. M. Correia, and D. B. Haddad, "Autonomous unmanned aerial vehicles in search and rescue missions using real-time cooperative model predictive control," *Sensors*, vol. 19, no. 19, p. 4067, 2019.

[15] D. Kim, R. N. Uma, H. B. Abay, W. Wu, W. Wang, and O. A. Tokuta, "Minimum latency multiple data muletrajectory planning in wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 13, no. 4, pp. 838–851, Oct. 2014.

[16] *Airborne Drones*. Accessed: Jan. 7, 2021. [Online]. Available: https://www.airbornedrones.co/natural-disasters/

[17] D. Lee, J. Zhou, and W. T. Lin, "Autonomous battery swapping system for quadcopter," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2015, pp. 118–124.

[18] Z. Liu, Z. Wang, D. Leo, and X. Liu, "QUADO: An autonomous recharge system for quadcopter," in *Proc. Int. Conf. Cybern. Intell. Syst.*, Oct. 2017, pp. 7–12.

[19] Z.-N. Liu, X.-Q. Liu, L.-J. Yang, D. Leo, and H. Zhao. (Oct. 2018). *An Autonomous Dock and Battery Swapping System Multirotor UAV*. [Online]. Available: https://www.researchgate.net/publication/325077351

[20] S. Huang, R. S. H. Teo, and K. K. Tan, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annu. Rev. Control*, vol. 48, pp. 147–164, 2019.

[21] D. Zhang, S. Sessa, R. Kasai, S. Cosentino, C. Giacomo, Y. Mochida, H. Yamada, M. Guarnieri, and A. Takanishi, "Evaluation of a sensor system for detecting humans trapped under rubble: A pilot study," *Sensors*, vol. 18, no. 3, p. 852, Mar. 2018.

[22] P. Rudol and P. Doherty, "Human body detection and geolocalization for UAV search and rescue missions using color and thermal imagery," in *Proc. IEEE Aerosp. Conf.*, Oct. 2008, pp. 1–8.

[23] E. Lygouras, N. Santavas, A. Taitzoglou, K. Tarchanidis, C. Athanasios Mitropoulos, and A. Gasteratos, "Unsupervised human detection with an embedded vision system on a fully autonomous UAV for search and rescue operations," *Sensors*, vol. 19, no. 16, p. 3542, 2019.

[24] I. Martinez-Alpiste, G. Golcarenarenji, Q. Wang, and J. M. A. Calero, "Search and rescue operation using uavs: A case study," *Expert Syst. Appl.*, vol. 178, Oct. 2021, Art. no. 114937.

[25] T. Erdelìe and T. Carié, "A survey on the electric vehicle routing problem: Variants and solution approaches," *J. Adv. Transp.*, vol. 2019, pp. 1–48, May 2019.

[26] J. C. S. Brandão and A. Mercer, "The multi-trip vehicle routing problem," *J. Oper. Res. Soc.*, vol. 49, no. 8, pp. 799–805, Aug. 1998.

[27] C. Archetti, D. Feillet, A. Mor, and M. G. Speranza, "An iterated local search for the traveling salesman problem with release dates and completion time minimization," *Comput. Oper. Res.*, vol. 98, pp. 24–37, Oct. 2018.

[28] R. Jothi and B. Raghavachari, "Approximating the k-traveling repairman problem with repairtimes," *J. Discrete Algorithms*, vol. 5, no. 2, pp. 293–303, 2007.

[29] Z. Xu, D. Xu, and W. Zhu, "Approximation results for a min-max location-routing problem," *Discrete Discrete Appl. Math.*, vol. 160, no. 3, pp. 306–320, Feb. 2012.

[30] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, Jun. 2006.

[31] M. Özlen and M. Azizoálu, "Multi-objective integer programming: A general approach for generating all non-dominated solutions," *Eur. J. Oper. Res.*, vol. 199, no. 1, pp. 25–35, Nov. 2009.

[32] R. Aringhieri, P. Landa, and S. Mancini, "A hierarchical multi-objective optimisation model for bed levelling and patient priority maximisation," in *Optimization and Decision Science: Methodologies and Applications*, A. Sforza and C. Sterle, Eds. Cham, Switzerland: Springer, 2017, pp. 113–120.

[33] G. Fancello, S. Mancini, C. Pani, and P. Fadda, "An emergency vehicles allocation model for major industrial disasters," *Transp. Res. Proc.*, vol. 25, pp. 1164–1175, Oct. 2017.

[34] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *Eur. J. Oper. Res.*, vol. 88, no. 3, pp. 464–474, 1996.

[35] A. Blum, S. Chawla, R. David Karger, T. Lane, A. Meyerson, and M. Minkoff, "Approximation algorithms for orienteering and discounted-reward TSP," *SIAM J. Comput.*, vol. 37, no. 2, pp. 653–670, 2007.

[36] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *J. Heuristics*, vol. 20, no. 3, pp. 291–328, 2014.

[37] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. Van Oudheusden, "Metaheuristics for tourist trip planning," in *Metaheuristics in the Service Industry*, K. Sörensen, M. Sevaux, W. Habenicht, and M. J. Geiger, Eds. Berlin, Germany: Springer, 2009, pp. 15–31.

[38] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," in *Proc. Symp. New Directions Recent Results Algorithms Complex.* Pittsburgh, PA, USA: Carnegie-Mellon Univ., 1976, pp. 1–6.

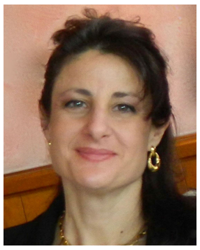
[39] S. Har-Peled, *Geometric Approximation Algorithms*. Providence, RI, USA: American Mathematical Society, 2011.

[40] A. Fotouhi, H. Qiang, M. Ding, and M. Hassan, "Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3417–3442, 4th Quart., 2019.

[41] A. Fotouhi, M. Ding, and M. Hassan, "DroneCells: Improving 5G spectral efficiency using drone-mounted flying base stations," *CoRR*, abs/1707.02041, pp. 1–5, Oct. 2017.

[42] A. Thomas Feo and G. C. Mauricio Resende, "Greedy randomized adaptive search procedures," *J. Global Optim.*, vol. 6, pp. 109–134, Oct. 1995.

[43] S. Mancini, "A combined multistart random constructive heuristic and set partitioning based formulation for the vehicle routing problem with time dependent travel times," *Comput. Oper. Res.*, vol. 88, pp. 290–296, Mar. 2017.



TIZIANA CALAMONERI graduated in mathematics, in 1992, and received the Ph.D. degree in computer science, in 1997. She is currently a Professor with the Department of Computer Science, Sapienza University of Rome, where she has been an Assistant Professor, since 2000, and an Associate Professor, since 2006. Her research interests include applications of graph algorithms, including problems arising from wired and wireless networks and from biology. She was the President of the Italian Chapter of the European Association for Theoretical Computer Science, from 2011 to 2017.

FEDERICO CORÒ (Member, IEEE) received the M.Sc. degree in computer science from the University of Perugia, Italy, in 2016, and the Ph.D. degree in computer science from the Gran Sasso Science Institute, L'Aquila, Italy, in 2019. From 2019 to 2020, he was a Postdoctoral Researcher at the Department of Computer Science, Sapienza University Rome, Italy. He is currently a Postdoctoral Researcher at the Missouri University of Science and Technology, USA. His research interests include several aspects of theoretical computer science, including combinatorial optimization, network analysis, and the design and efficient implementation of algorithms.

SIMONA MANCINI graduated in mathematical engineering from the Politecnico di Torino, in 2007. She received the Ph.D. degree in computers and systems engineering from the Politecnico di Torino, in 2011. She worked as an Assistant Professor in operations research at the University of Cagliari, from 2015 to 2018, and an Universitaet Assistentin (an Assistant Professor) at the University of Klagenfurt, from 2020 to 2021, where she has been in charge of several master courses in operations management and logistics. She is currently a Tenure-Track Assistant Professor in computers science at the University of Eastern Piedmont, Italy.