A Multi-Trip Task Assignment for Early Target Inspection in Squads of Aerial Drones

Novella Bartolini, *Senior Member, IEEE,* Andrea Coletta, *Student Member, IEEE,*
Gaia Maselli, *Member, IEEE,* Ala' Khalifeh, *Member, IEEE.*

Abstract—Fleets of cooperative drones are a powerful tool in monitoring critical scenarios requiring early anomaly discovery and intervention. Due to limited energy availability and application requirements, drones may visit target points in consecutive trips, with recharging and data offloading in between. To capture timeliness of intervention and prioritize early coverage, we propose the new notion of Weighted Progressive Coverage, which is based on the definition of time dependent weights. Weighted progressive coverage generalizes classic notions of coverage, as well as a new notion of accumulative coverage specifically designed to address trip scheduling. We show that weighted progressive coverage maximization is NP-hard and propose an efficient polynomial algorithm, called Greedy and Prune (GaP), with guaranteed approximation. By means of simulations we show that GaP performs close to the optimal solution and outperforms a previous approach in all the considered performance metrics, including coverage, average inspection delay, energy consumption, and computation time, in a wide range of application scenarios. Through prototype experiments we also confirm the theoretical and simulation analysis, and demonstrate the applicability of our algorithm in real scenarios.

Index Terms—trajectory planning, UAV, drones, task assignment, vehicle routing.

1 Introduction

Fleets of aerial drones are a powerful tool for monitoring safety critical scenarios, performing damage assessment and search and rescue operations. They are increasingly deployed in post disaster scenarios for several purposes, including quick spot inspection, drone assisted cellular communications, airdropping of food and medicines to people in need [1], [2].

Despite the large use of squads of aerial drones in many critical settings, coordination and control is mostly left to human personnel supervising the flight operations. Automated trajectory planning is an open challenge, especially for those applicative scenarios where early inspection is of utmost importance. For instance, in critical scenarios where drones are looking for survivors, or aim at distributing medicines or water to trapped or disabled humans or animals, timeliness of intervention is a strict requirement. Thus, the benefit of using multiple aerial devices capable of sampling multiple targets in parallel, and ensuring the best tradeoff between monitoring accuracy and rapidity of intervention is clear. However, planning drone trajectories is still a major challenge under several points of view, including autonomy, cooperation, coordination, and control.

Previous approaches to trajectory planning for squads of drones mainly consider a one-trip per drone assignment, with the purpose of maximizing the number of inspected targets, or minimizing the task completion time. However, as aerial drones usually have limited energy and storage availability, they may require multiple trips to provide complete monitoring coverage, with battery replacement/charging and data offloading in between. Unlike previous work we address the problem of designing the trip trajectories, so as to schedule the most inspections in the earliest trips, under the drone energy constraints. Moreover, we consider heterogeneous drones (e.g., with different batteries and energy consumption curves), that may depart from different depots.

For the purpose of evaluating and optimizing the capability of an algorithm to provide early monitoring, we define a novel metric called *Weighted Progressive Coverage* (WPC). The optimization of this metric allows to jointly address the problems of target coverage and task scheduling. The definition of WPC generalizes the classic notion of total coverage and also allows to give more value to tasks scheduled in the earlier rounds. The *accumulative coverage* metric is also introduced as a special instance of WPC, which assigns linearly decreasing values to the tasks executed in progressive rounds. We show how the optimization of the accumulative coverage metric successfully prioritizes early target inspection and minimizes the average inspection delay caused by inter-trip operations: battery replacing/recharging and/or data offloading can have a significant impact on the overall mission time.

We contribute an analytic formulation of the WPC maximization problem as an Integer Linear Programming (ILP) model and show its NP-hardness. Experiments confirm that the optimization problem has prohibitive execution times for problem instances of average to large size, requiring many hours of computations. For this reason we propose a very efficient polynomial algorithm with guaranteed approximation of the optimal solution.

We compare our proposal to several previous approaches, through extensive simulations and real field experiments, showing that our approach outperforms pre-

- *N. Bartolini, A. Coletta and G. Maselli are with the Department of Computer Science, Sapienza University of Rome, Italy. E-mail: {coletta,bartolini,maselli}@di.uniroma1.it*
- *A. Khalifeh is with the Computer Engineering Department, German Jordan University, Jordan. E-mail: ala.khalifeh@gju.edu.jo*

vious algorithms in all the relevant performance aspects including coverage, average inspection delay, as well as energy cost and computation time.

In summary, the main contributions of this paper are:

- We formulate a novel performance metric, called *Weighted Progressive Coverage*, to measure the capability of a trajectory planning algorithm to provide early target inspection, and formulate the related optimization problem. We show that under appropriate settings of the weight function, weighted progressive coverage translates into traditional metrics of coverage, as well as into a new notion of accumulative coverage which clearly reflects early coverage capabilities.
- We formulate the problem of maximizing weighted progressive coverage as an ILP model and prove its NP-hardness. By means of experiments we show that such optimization has prohibitive computation times even for small problem instances, which precludes its application in safety critical scenarios, and motivates the need of polynomial time heuristics.
- We propose a novel greedy-and-prune polynomial algorithm for approximating the optimal solution of the aforementioned problem. We prove that the above algorithm has a constant factor approximation of $1/2$.
- We study the proposed approaches in an extensive range of operational settings through simulations, and we confirm the analysis via real field experiments in a test-bed implementation. The study shows that the performance of our approach is close to the optimal, and considerably better than previous solutions in all the performance metrics, including coverage, inspection delay, energy consumption and computation time.

This paper extends a previous conference contribution [3].

2 Related Work

Previous works addressing path planning management for teams of UAVs proposed several approaches including: graph algorithms, optimization problems, genetic and machine learning algorithms. A common requisite for all the approaches is to determine one or multiple routes satisfying given requirements related to the visit of a set of points of interest in an area.

The first approach considers variants of the Traveling Salesman Problem (TSP) for multiple vehicles [4]. Specifically, the multi-traveling salesmen problem, and its variant, the vehicle routing problem (VRP) [5], aim at finding the shortest, or lowest cost paths for the mobile devices to visit all the points of interest under a variety of constraints such as an upper bound on maximum cost for each trajectory or on the maximum number of traversed points of interest; the multi-repairmen problem aims at covering all the points of interest and minimizing the average visit delay of the points [6], [7], [8]; finally, the Capacitated Vehicle Routing Problem (CVRP) [9] aims at visiting points of interest, while delivering goods under limited vehicle capacity.

The works [10] and [11] generalize the multi-traveling salesmen problem for multi-UAV path planning for reconnaissance and surveillance contexts. However, they do not consider the limited energy availability of the drones, an inevitable constraint of any battery powered devices. In particular, in [11] the authors propose a path planning algorithm for multiple UAVs which aims at minimizing the mission completion time in search-and-reconnaissance operations. In Section 6.2 we design an energy-constrained variant (shortly called *TCTPA*) of this approach, which is used in our experiments for performance comparisons.

Another line of research considers the use of linear programming tools to guide decision making in mobile vehicles operations [12], [13], [14]. The work [12] proposes a novel system for wildfire monitoring using a fleet of UAVs to provide on-demand fast services, through a distributed leader-follower coalition algorithm which aims at providing full coverage of the area in a timely manner, while minimizing the drones and the energy consumption. Despite the similarity with the problem addressed in our paper, the approach proposed in [12] does not explicitly address limited resource scenarios as we do in our paper to provide multiple flight rounds (with battery recharging or replacement) when the number of drones is inadequate to cover all the target points within a region in a single trip. The work in [13] studies the problem of trajectory planning for fleets of UAVs in search and rescue (SAR) operations, especially in maritime accidents. In this work the UAVs can be recharged at different stations to complete the mission. The authors propose a mixed integer linear program (MILP) to generate efficient search and rescue operation plans. The main limitation is scalability in terms of number of drones and points of interest due to the high computational complexity of the proposed solutions. The work in [14] considers a different scenario where drones are employed to provide network connectivity to a group of users on the ground. Therefore the goal of trajectory planning is to maximize users' throughput, which is different from our objective.

Other approaches leverage genetic algorithms to design the trajectories of multiple UAVs in different missions [15] [16] [17]. The work in [15] considers a fleet of drones with limited capabilities. The work solves a capacitated Vehicle Routing Problem (CVRP) for multiple drones by using a genetic algorithm providing selection, mutation and crossover. However, the solution is efficient only for scenarios with few points of interest; as the number of points to be visited increases, the required computational power increases significantly. In intelligence transportation systems, the work in [16] aims at finding time-optimal paths for multiple UAVs such that they collectively visit some target areas when they fly from a starting point to a final location. However, differently from ours, this work considers only a single-depot and a single trip trajectory for each drone.

We also mention a few solutions that are based on artificial intelligence. The work in [18] proposes an unsupervised learning algorithm for solving K-DTSP with Neighborhoods (i.e. a multi-vehicle variant of TSP where a target is considered visited provided that a drone trajectory traversed an area surrounding it) with curvature constraints. The work in [19] uses a distributed reinforcement learning algorithm to find drone paths in a dynamic scenario where the final destination for each drone is not fixed a priori. These works lack of any performance guarantee with respect to the optimal solution; moreover learning approaches usually require significant time to converge to a steady state.

Finally, there is a large number of works on mission control for ground mobile robots, which are not suitable for the study of UAV task assignment. Assumptions, requirements and constraints for UAVs are different from those for mobile robots. Aerial drones have lightweight payloads and shorter operative time, often requiring multiple trips; moreover, they consume energy also when hovering and have higher speed. The work by Setter et al. [20] addresses the problem of coordinating a squad of mobile robots to perform rendezvous operations within device energy constraints. The work by Popescu et al. [21] tackles the robot patrolling problem to repeatedly cover a set of targets during a mission. The works by Yazici et al. [22], and Mei et al. [23] address the problem of area coverage, with homogeneous devices. Some of our experiments also study an extension of our approach to the optimization of area coverage by designing a variant of the target coverage problem. For these experiments we also adopt the algorithm *Sweep*, inspired by [23], as a benchmark for comparisons in this problem setting.

To the best of our knowledge, our work is the first that tackles the drone path planning problem into a general scenario requiring early target inspection. We consider multiple-round exploration and maintenance phases in order to work also in under-provisioned settings. (e.g., where the drone batteries are not sufficient to cover all the targets in one single trip, therefore they are allowed to return to the depot station to have their battery replaced or recharged). We propose a novel optimization problem, where the flight dynamics, energy consumption, and restricted flight zones are directly incorporated in the input and early inspection is explicitly measured and optimized; finally, we propose an efficient polynomial algorithm with guaranteed performance, evaluated against other proposals by means of both simulations and prototype experiments.

This work extends a previous conference version [3]. With respect to the previous work the major enhancements are the following: in Section 4 we introduce a novel problem formulation which directly computes trajectories and we provide a related algorithm for trajectory design in Section 5.3; we present real field experiments in Section 7, with an in-depth discussion on the key implementation challenges. In the Section 7 we also extend the evaluation to the different operation scenario requiring area coverage as opposed to target coverage.

3 Problem Formulation

We consider the scenario in which a squad of aerial drones aims at inspecting a given set of target points Ψ in the region of interest. These points may be the known locations of survivors of a catastrophe, requiring medicines or water to be dropped from aerial drones, or more in general they may be areas of suspected anomalies, requiring immediate surveillance and local inspection. Figure 1 shows an example of the system with two different depots, d_1 and d_2, a set of 6 heterogeneous drones, and several critical points to visit, i.e. the red markers in the figures. The goal of our system is to compute — at the mission start — the multi-trip trajectories that the squad should follow in order to visit all points of interest.

Fig. 1. System scenario

Let $\mathcal{U}$ be the set of aerial vehicles forming the squad, and let d_u be the home depot of drone $u \in \mathcal{U}$, i.e. the point of the region of interest from which the drone departs, and where it is recollected, its data are offloaded, and batteries are recharged at the end of each trip. It is important to notice that the battery of a drone imposes a limitation on the single trip duration. For instance, the DJI AGRAS T16 [24] drone, used to nebulize water and medicines on crop fields for professional agricultural applications, can only fly for 18 minutes in hovering mode without payload, considering basic battery equipment. The flight time may be instead as low as 10 minutes when the drone is fully loaded (i.e., with a payload of about 16 Kg), an amount of time which becomes even smaller if the drone performs complex flight maneuvres. In our experimental test-bed described in Section 7, we use DJI Flame Wheel F550 [25] drones, equipped with a LIPO battery providing 3500mAh 25C, which allows the drone to fly at 5 m/s for about 7 minutes with no additional payload.

3.1 Progressive coverage metrics

We consider a progressive execution of the monitoring activity of the drone squad in consecutive rounds $n = 1, 2, \ldots, N$, where new target points are inspected at each new round. A round based execution is meant to consider recharging and offloading at the depot between consecutive rounds. We denote with $[N]$ the set $\{1, 2, \ldots, N\}$. The variables $\delta_i(n) \in \{0, 1\}$ represent the coverage status of target i at round $n \in [N]$ for any $i \in \Psi$. If target i is visited exactly at round n, then $\delta_i(n) = 1$, otherwise $\delta_i(n) = 0$. To avoid repeated coverage of a single point i, we impose $\sum_{n=1}^{N} \delta_i(n) \leq 1$.
We denote with

$$\delta(n) \triangleq \sum_{i \in \Psi} \delta_i(n) \tag{1}$$

the amount of target points covered exactly at round n, that we shortly call *round coverage*.

We also define a round based utility function $w(n)$ reflecting non increasing utilities for each round, such that $w(n_1) \geq w(n_2) \geq 0$, when $n_1 < n_2$ for $n_1, n_2 \in [N]$.

We now introduce a novel metric $\mathcal{W}(n)$ called *Weighted Progressive Coverage (WPC)*, defined as the weighted average of the round coverage, calculated over the first n rounds:

$$\mathcal{W}(n) \triangleq \sum_{k=1}^{n} w(k) \cdot \delta(k). \tag{2}$$

The specific setting of the weights $w(k)$ at round k is meant to give different priorities to the earlier trips, considering one trip per drone at each round.

It is easy to see that the *total coverage in n rounds*, hereby denoted with $\Delta(n)$, is obtained by setting $w(k) = 1,\ \forall k = 1, \ldots, N$, in Equation (2) as follows:

$$\Delta(n) \triangleq \sum_{k=1}^{n} \delta(k), \tag{3}$$

which represents the number of targets covered either at round n or at any round before n. Hence total coverage is a special case of WPC.

We now define a novel coverage metric which prioritizes early coverage of target points, called *accumulative coverage* $\mathcal{A}(n)$ *at round* n:

$$\mathcal{A}(n) \triangleq \sum_{k=1}^{n} \Delta(k). \tag{4}$$

We observe that the accumulative coverage metric is also a special case of WPC, obtained by setting the values of the weights $w(k)$, $k = 1, \ldots, n$, as explained in the following observation.

Observation 3.1. *The accumulative coverage function $\mathcal{A}(n)$ at round n is a linear combination of the single round coverage variables which can be expressed as:*

$$\mathcal{A}(n) = \sum_{k=1}^{n} (n - k + 1) \cdot \delta(k). \tag{5}$$

Therefore the accumulative coverage function at round n is a special case of WPC obtained by setting the values $w(k) = (n - k + 1)$, in Equation (2).

Proof. By simple algebraic passages, we see that

$$\mathcal{A}(n) = \sum_{k=1}^{n} \Delta(k) = \sum_{k=1}^{n} \sum_{j=1}^{k} \delta(j) = \sum_{k=1}^{n} (n - k + 1) \cdot \delta(k). \quad \square$$

Table 1 summarizes the four coverage definitions.

TABLE 1
Summary of notations

Notation	Description
$\delta(n)$	*Round coverage* at round n - Eq. 1
$\mathcal{W}(n)$	*Weighted Progressive Coverage (WPC)* at round n - Eq. 2
$\Delta(n)$	*Total coverage* at round n - Eq. 3
$\mathcal{A}(n)$	*Accumulative coverage* at round n - Eq. 4

WPC and, more specifically, its accumulative coverage variant, reflects the capability of a trajectory planning algorithm to prioritize coverage in the early rounds.

As an example, let us consider a set of 30 targets to be visited within $N = 5$ rounds. The plots of Figure 2 show, in row order, the value of round, total, and accumulative coverage. The figure considers two different cases of task assignment, where the available drones execute different trips at each round $n \in [N]$, as evidenced by the first row, which highlights the covered targets per round, i.e., round coverage. We see that the value of total coverage, shown in the second row, is the same for the two cases at the end of the first five rounds, as both the executions ended the fifth round with complete coverage of the set of targets. Measuring the the total coverage metric at the end of the mission does not capture the difference between the two executions as it only considers what was covered at the end of the flight schedule. In contrast, the accumulative coverage value, shown in the third row, properly captures the fact that the trips designed for the second case provide higher round coverage in the early rounds. The accumulative coverage value at the end of the fifth round is in fact higher in the second case than in the first case of execution. This is due to the higher weights assigned to the initial rounds in the computation of the accumulative coverage value, which reflect the prioritization of early inspections.

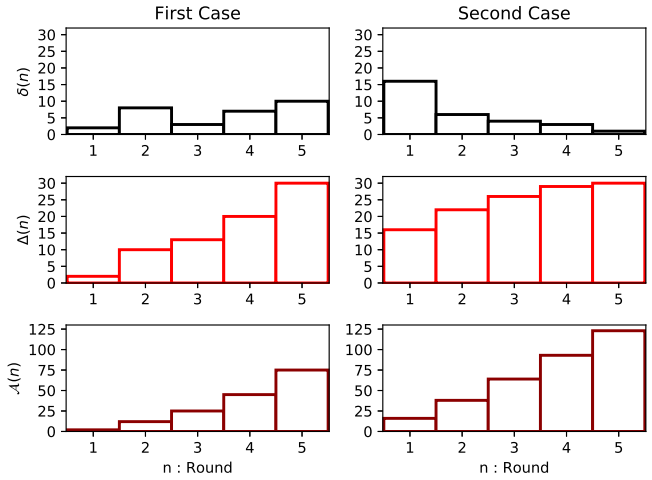


Fig. 2. Round-, total-, and accumulative-coverage with two different choices of trajectories for multiple rounds of flight.

Consider now N rounds, and a setting where all the target points of Ψ are covered in a progressive, round based inspection, namely $\sum_{k=1}^{N} \delta(k) = |\Psi|$.

Let $\tau_i(N)$ be the round at which target i is visited, hereby called the *inspection delay* of target $i \in \Psi$ in an N-rounds progressive solution. It holds that:

$$\tau_i(N) \triangleq \sum_{k=1}^{N} k \cdot \delta_i(k). \tag{6}$$

We denote with $D(N)$ the *average inspection delay*:

$$D(N) = \sum_{i \in \Psi} \tau_i(N) / |\Psi|. \tag{7}$$

Theorem 3.1. *For any progressive coverage solution $\delta_i(k)$, with $i \in \Psi$ and $k \in [N]$, it holds that:*

$$\mathcal{A}(N) = |\Psi| \cdot [N + 1 - D(N)], \tag{8}$$

where $\mathcal{A}(N)$ and $D(N)$ are defined in Equations (5) and (7), respectively. It follows that any progressive coverage solution which guarantees complete coverage of the set Ψ and maximizes the accumulative coverage function $\mathcal{A}(N)$, also minimizes the average inspection delay $D(N)$.

Proof. By applying Observation 3.1 we have that

$$\mathcal{A}(N) = (N + 1) \cdot \sum_{k=1}^{N} \delta(k) - \sum_{k=1}^{N} k \cdot \delta(k) =$$

$$= (N+1)\cdot|\Psi| - \sum_{k=1}^{N}\sum_{i\in\Psi} k\cdot\delta_i(k) = (N+1)\cdot|\Psi| - \sum_{i\in\Psi}\tau_i(N) =$$
$$= |\Psi|\cdot[(N+1) - D(N)]\,. \quad \square$$

Discussion on the setting of the number of rounds N: We clarify that the setting of N — the maximum number of rounds for inspecting the target points in the field of interest — typically responds to application requirements, for example if drones are supposed to operate during daylight hours. The setting of N affects the total duration of the target inspection mission. Assuming that each drone has enough energy to inspect at least a target of Ψ at each round, an upper bound on the setting of N is $|\Psi|/|\mathcal{U}|$, which is the maximum number of rounds required to ensure completion of the target covering mission (i.e., total coverage of the target points).

Discussion on the round length and synchronization: The WPC metric gives the same weight to the inspection of different targets when they occur at the same round. This requires that (1) the time between two consecutive rounds is at least as long as a single round duration to ensure that the time between two inspections occurring in the same round is always shorter than the time between inspections occurring in consecutive rounds; (2) inspection rounds executed by different drones have a similar length, which ensures that round based utilities are monotonically decreasing with respect to the target inspection times, regardless of the drone being considered. However we notice that the first assumption is verified in most application scenarios. In fact, maintenance operations between consecutive trips (e.g., battery recharging and data offloading) typically require a time that is a factor of 3 to 8 times the maximum flight duration. In the experimental test-bed considered in Section 7, this factor is about 10. The second assumption is also valid whenever drones have roughly homogeneous flight capabilities and their trips are terminated only when the residual energy is not sufficient to inspect any more targets, with the exception of the last round. In most operative cases, with battery powered copters, the above assumptions are verified as the inter-round operation time usually dominates the drone flight duration. If these assumptions are not verified for a specific problem instance, the WPC metric definition may be revised to consider utilities which decrease with the actual target inspection time.

4 WPC OPTIMIZATION

The problem of maximizing WPC or any of its variants, including total and accumulative coverage, can be formulated as an Integer Linear Programming (ILP) model, as detailed in Problem 1. The objective function (a) of the problem is the expression of WPC defined in Equation 2, where the integer variable $\delta(n)$ denotes the amount of targets covered at round n, i.e., the round coverage defined by Equation 1, and in constraint (h) of the problem. The binary variables $x_{ij}^u(n)$, with $i,j \in \Psi \cup \{d_u\}$, $u \in \mathcal{U}$, $n \in [N]$, reflect the decision to let drone u traverse the route between the locations of target i and j, exploring them in a sequence ($x_{ij}^u(n) = 1$), or not ($x_{ij}^u(n) = 0$), at the n-th trip. Based on these variables, constraint (b) imposes that a target be traversed the same number of times in entry and exit direction. Sub-tour elimination is obtained via the MTZ technique proposed by Miller, Tucker and Zemlin in [26], with the constraints (c-d), where the auxiliary integer variables $z_i^u(n)$ represent the ordinal position of target i in the trajectory of drone u at round n. Constraint (e) imposes an upper bound b_u on the maximum energy expenditure of drone u at each round, where b_u is the drone battery availability (in energy units), and ϕ_{ij}^u is a constant value, reflecting the energy cost, for drone u to travel from target i to target j and inspect target j (we refer to Section 4.2 for a discussion on the setting of the parameters ϕ_{ij}^u). Constraint (f) relates the value of the target coverage variable $\delta_i(n)$ to the values of the variables $x_{ij}^u(n)$, and determines the decision to cover target $i \in \Psi$ at round n. Constraint (g) precludes redundant coverage of the same target points, constraint (h) relates the round coverage $\delta(n)$ with the values of the variables $\delta_i(n)$ denoting coverage of individual targets, while constraints (i) define the variable domains.

$$\begin{array}{ll}
\max \mathcal{W}(N) \triangleq \sum_{i=1}^{N} w(i)\cdot\delta(i) & (a)\\
s.t., \forall n = 1,\ldots,N & \\
\sum_{i\in\Psi\cup\{d_u\}} x_{ij}^u(n) = \sum_{k\in\Psi\cup\{d_u\}} x_{jk}^u(n), \forall j\in\Psi, u\in\mathcal{U} & (b)\\
z_j^u(n) - z_i^u(n) \geq x_{ij}^u(n) + |\Psi|\cdot(x_{ij}^u(n)-1), \forall u\in\mathcal{U}, i,j\in\Psi & (c)\\
z_{d_u}^u(n) = 0;\ z_i^u(n) \in \{1,\ldots,|\Psi|\}, \forall i\in\Psi, u\in\mathcal{U} & (d)\\
\sum_{ij\in\Psi\cup\{d_u\}} \phi_{ij}^u \cdot x_{ij}^u(n) \leq b_u, \forall u\in\mathcal{U} & (e)\\
\delta_i(n) = \sum_{u\in\mathcal{U}, j\in\Psi\cup\{d_u\}} x_{ij}^u(n), \forall i\in\Psi & (f)\\
\sum_{n=0}^{N} \delta_i(n) \leq 1, \forall i\in\Psi & (g)\\
\delta(n) = \sum_{i\in\Psi}\delta_i(n) & (h)\\
x_{ij}^u(n)\in\{0,1\}, \delta_i(n)\in\{0,1\}, \forall i,j\in\Psi, u\in\mathcal{U} & (i)
\end{array}$$

Problem 1. WPC Optimization (edge-related variables)

4.1 Formulation with restricted sets of paths

The solution space defined by the constraints of Problem 1 includes all the cyclic trajectories $\mathcal{C}_u$ that each drone u can traverse within its battery limitation, starting from its depot d_u. In Problem 2 we provide an equivalent formulation, considering trajectory related variables $z_p^u(n) \in \{0,1\}$, to denote the decision to assign the cyclic path $p \in \mathcal{C}_u$ to drone $u \in \mathcal{U}$ at round $n \in [N]$. Constraint (b) defines the variables $\delta_i(n)$ as a function of the trajectory related variables $z_p^u(n)$, constraint (c) imposes that each drone traverses at most one of the cyclic trajectories at any given round n, constraint (d) precludes redundant coverage, and the remaining constraints (e-f) define the domain of the decision variables of the problem.

$$\begin{array}{ll}
\max \mathcal{W}(N) \triangleq \sum_{i=1}^{N} w(i)\cdot\delta(i) & (a)\\
s.t. & \\
\delta_i(n) = \sum_{u\in\mathcal{U}, p\in\mathcal{C}_u: i\in p} z_p^u(n), \forall i\in\Psi,\ \forall n\in[N] & (b)\\
\sum_{p\in\mathcal{C}_u} z_p^u(n) \leq 1,\ \forall u\in\mathcal{U},\ \forall n\in[N] & (c)\\
\sum_{n=1}^{N} \delta_i(n) \leq 1, \forall i\in\Psi & (d)\\
z_p^u(n)\in\{0,1\},\ \forall u\in\mathcal{U}, p\in\mathcal{C}_u,\ \forall n\in[N] & (e)\\
\delta_i(n)\in\{0,1\},\ \forall i\in\Psi,\ \forall n\in[N] & (f)
\end{array}$$

Problem 2. WPC Optimization (path-related variables)

The problem is clearly NP-hard as it generalizes the Travelling Salesman Problem (TSP) [27]. The number of variables of Problem 2 grows with the number of cyclic trajectories that can be defined on Ψ. Computing all these trajectories

is time-consuming, and can become the bottleneck of the entire trajectory planning problem.

Moreover, it must be noted that in practice not all the trajectories are equally viable due to no fly-zones, energy constraints, and limited maneuverability of drones [28].

Provisioning the list of viable trajectories for each drone is in itself an interesting research problem. The work by Milan et al. [29] discusses the inherent non linearity of this problem and addresses the generation of spline trajectories by means of non-linear models which take account of mechanical constraints. Trajectories may be generated by means of clustering algorithms, including density based clustering [30], agglomerate hierarchical clustering [31], and spectral clustering [32], just to mention those that may capture the multivariate nature of the target geographical distribution.

In the following we consider a set of *candidate trajectories* for each drone $\chi_u(b_u, d_u)$, as the set of trajectories that drone $u \in \mathcal{U}$ can use ($\chi_u \subseteq \mathcal{C}_u$). The definition of this set depends on the battery limitation b_u of drone u, on the adopted energy consumption model, and on the location of the related depot d_u, as well as on the other limitations discussed above. We denote $\chi \triangleq \cup_{u \in \mathcal{U}} \chi_u$.

Notice that the WPC optimization problem under restricted sets of paths χ_u, for $u \in \mathcal{U}$, requires a more general definition of the coverage variables $\delta_i(n)$ to take account of the fact that the optimization problem could select paths with overlapping coverage of some targets. This is because the use of restricted paths does not ensure the existence of a full coverage solution composed of disjoint paths only, and we cannot exclude the selection of paths traversed by different drones and/or at different rounds, containing the same targets. In Section 5.2 we discuss a pruning technique to be adopted to post-process a solution to the task assignment problem based on restricted paths, for removing redundantly covered target points without affecting weighted coverage.

For the purpose of redefining Problem 2 under a restricted set of paths χ, we generalize the definition of the variables $\delta_i(n)$ given in Section 3.1 to this new setting. Therefore, we define $\delta_i(n) = 1$ if n is *the first round* in which i is covered by one or more drones, and $\delta_i(n) = 0$ otherwise. The definition of $\tau_i(N)$ is generalized accordingly, so that $\tau_i(N)$ corresponds to the number of *the first round* at which target i is traversed for the first time.

Naturally, the generalized variables $\delta_i(n)$ and $\tau_i(N)$ will produce the same values described in Section 3.1 when adopted in a scenario where the set of paths is unrestricted.

In agreement with Equation (6), in which we defined $\tau_i(N)$ for the case of unrestricted path sets, we define

$$\tau_i(N) \triangleq \min\{n \in [N] : \exists (p, u) \text{ with } p \in \chi_u, i \in p, z_p^u(n) = 1\},$$

when i is covered, otherwise we set $\tau_i(N) = N + 1$.

Observation 4.1. *The values of $\delta_i(n),\ i \in \Psi$, are uniquely determined from the values of the variables $z_p^u(n),\ u \in \mathcal{U},\ p \in \chi_u$, defining a round based trajectory assignment. Then $\delta_i(n) = 1$ if $n = \tau_i(N)$ and $\delta_i(n) = 0$ otherwise, for $n \in [N]$.*

Problem 2 must be modified to take account of restricted sets of paths, and of the new variable definitions we have just exposed. The new optimization problem, under restricted sets of paths, is the following Problem 3, where we replaced constraint (b) of Problem 2 with new constraints (b1) and (b2), as follows.

$$\begin{array}{ll}
\max \mathcal{W}(N) \triangleq \sum_{i=1}^{N} w(i) \cdot \delta(i) & (a) \\
s.t.,\ \forall n \in [N] & \\
\delta_i(n) \geq \sum_{u \in \mathcal{U}} \sum_{p \in \chi_u : i \in p} \frac{z_p^u(n)}{|\mathcal{U}|} - N \cdot \sum_{k=1}^{n-1} \delta_i(k),\ \forall i \in \Psi & (b1) \\
\delta_i(n) \leq 1 - \sum_{k=1}^{n-1} \frac{\delta_i(k)}{(n-1)},\ \forall i \in \Psi & (b2) \\
\sum_{p \in \chi_u} z_p^u(n) \leq 1,\ \forall u \in \mathcal{U} & (c) \\
z_p^u(n) \in \{0, 1\},\ \forall u \in \mathcal{U}, p \in \chi_u & (d) \\
\delta_i(n) \in \{0, 1\},\ \forall i \in \Psi & (e)
\end{array}$$

Problem 3. WPC optimization (restricted sets of paths).

The metric $\delta(n) = \sum_{i \in \Psi} \delta_i(n)$ represents the *number of newly covered targets* at round n. It generalizes the definition of $\delta(n)$ given in Section 3.1 for the case of unrestricted sets of paths.

4.2 Energy Models

Problem 1 abstracts the details of the energy model of a drone u by introducing the parameters related to the edge cost ϕ_{ij}^u, which include the costs of travelling from target i to target j, and the inspection cost at target j.

With a similar approach, Problems 2 and 3 also incorporate the energy limitation b_u of a drone u into the definition of the set of feasible trajectories $\mathcal{C}_u$ and χ_u, i.e., trajectories that each drone can traverse without depleting its battery before the end of trip.

By making the assumption that energy consumption is only related to inspection tasks and trajectories, this approach neglects environmental conditions, e.g., varying wind conditions. However, in both the formulations, the abstraction has the advantage of incorporating the energy model in linear terms of the ILP formulation. Many different energy models can be used to parametrize our problem formulation, e.g., those proposed by Goss et al. [33], while still keeping the optimization model simple and linear.

Moreover, the proposed approach has the advantage of considering drone heterogeneity, as both the parameters ϕ_{ij}^u of Problem 1 and the setting of feasible trajectories $\mathcal{C}_u$ and χ_u of Problems 2 and 3, respectively, can be calculated on the basis of device specific energy models.

Since no unique energy model can be general enough to capture the peculiarities of any specific device, in our experiments we adopt a simple linear model for which energy consumption is proportional to the travelled distance when the drone moves, and to the flight time when the drone hovers to inspect a target. This model is discussed in Section 6.1.

5 Efficient approximation algorithms for WPC optimization

The definition of restricted sets of paths χ_u for each drone $u \in \mathcal{U}$ significantly reduces the size of the feasible region of the WPC maximization problem, i.e., it reduces the number of cycles to be considered as potential trajectories. Theorem 5.1 shows that, despite these simplifications, the problem is still NP-hard, and motivates research for polynomial time

heuristics with good approximation of the optimal, within the constrained sets of trajectories.

Theorem 5.1. *Problem (3) is NP-hard.*

Proof. Any instance of the Max-Coverage problem can be reduced to an instance of Problem 3 in polynomial time. Consider a collection $\mathcal{S}$ of sets of elements of $\mathcal{T}$, and an integer value m. The Max-Coverage problem requires finding m elements $S_1, S_2, \ldots, S_m$ of $\mathcal{S}$ such that $\cup_{i=1}^{m} S_i$ is maximized. We can build an instance of our WPC problem by considering all the elements of $\mathcal{T}$, as target points, therefore $\Psi = \mathcal{T}$, and one only drone u, $|\mathcal{U}| = 1$ with depot d_u, flying for $N = m$ rounds. The restricted set of paths χ_u associated to drone u is then formulated as follows: for any $S \in \mathcal{S}$ we consider a path $p_S \in \chi_u$ including all the nodes of S and the depot d_u in a cyclic trajectory (any cycle fits our needs). We then set the energy availability of drone u to b_u such that b_u is sufficient to complete any trajectory p_S for any $S \in \mathcal{S}$. Finally, we set the weights $w(n) = 1$, for each $n = 1, \ldots, m$. Any solution to Problem 3 maximizes the number of elements of $\mathcal{T}$ covered by m sets of $\mathcal{S}$. The hardness of Problem 3 derives from the hardness of Max-Coverage. □

The hardness of Problem 3 motivates us to seek for efficient suboptimal solutions with guaranteed performance. To this end, we identify properties of the set of constraints and of the objective functions that allow for easy approximation. In the following we assume that the sets of feasible trajectories χ_u is known in advance, for each drone $u \in \mathcal{U}$, and defines the problem instance. In Section 5.3 we discuss efficient techniques for generating sets of feasible trajectories.

5.1 Trajectory Assignment as Matroid Optimization

We recall the following concepts from combinatorial optimization.

Definition 5.1 (Matroid [34])**.** *A matroid $\mathbb{M}$ is a pair $(E,\ I)$, where E is a finite ground set and $I \subseteq 2^E$ a non-empty collection of subsets of E, with the following properties:*

1) *$\forall A \subset B \subseteq E$, if $B \in I$, then $A \in I$;*
2) *$\forall A, B \in I$ with $|B| > |A|$, $\exists x \in B \backslash A$ such that $A \cup \{x\} \in I$.*

Definition 5.2 (Partition Matroid [35])**.** *A matroid $\mathbb{M} = (E, I)$ is a* partition matroid *if E is partitioned into* disjoint *sets E_1, E_2, ..., E_m and, for some given integers b_1, ..., b_m, $0 \leq b_i \leq |E_i|$,*

$$I = \{X \subseteq E \ :\ |E_i \cap X| \leq b_i,\ for\ i = 1, 2, \ldots, m\}.$$

Definition 5.3 (Monotone submodular function [34])**.** *Given a finite ground set E and a function $f : 2^E \to \mathbb{R}$,*

- *f is monotone if $\forall A \subset B \subseteq E$, $f(A) \leq f(B)$;*
- *f is submodular if $\forall A \subset B \subseteq E$ and $e \in E \setminus B$, $f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B)$.*

By proving that our objective function is monotonically increasing and submodular, and that our constraints form a matroid, we can apply results from combinatorial optimization to define algorithms with known approximation guarantee. More specifically, we recall the following result on the greedy approach[1].

Theorem 5.2 ([37])**.** *Consider the maximization of a set function $f : 2^E \to \mathbb{R}$ over a collection $I \subseteq 2^E$ of sets. We denote with f^* the optimal value, and with f^g the value achieved by the greedy algorithm. If $\mathbb{M} = (E,\ I)$ is a matroid and f is monotone and submodular, then $f^g \geq f^*/2$.*

We show that our trajectory assignment problem can be cast as the problem of maximizing a set function under matroid constraints. In fact, under the assumption that each drone u may be activated a given number of rounds at most equal to k_u, with full battery charge b_u, the constraints of Problem 3 define a set of feasible solutions that can be mapped to a partition matroid.

Each drone selects up to k_u cycles on a round basis, for a total of maximum N rounds, where $N = \max_{u \in \mathcal{U}} k_u$. For simplicity, we hereby consider $k_u = N,\ \forall u \in \mathcal{U}$. Notice that the round-based cycle selection does not imply that the drone activities must be synchronized. Nevertheless as a first study we consider the round based weight decreasing in the number of elapsed rounds (namely, targets visited in the first round have the largest weight, regardless of the time taken by their drone to complete the round).

In the following lemma we show that the feasible region of Problem 3 can be mapped to a matroid.

Lemma 5.1. *The solution space of Problem 3 can be modelled by means of the pair $M = (\mathcal{E}, \mathcal{I})$, defined as follows:*

- *$\mathcal{E}$ is a ground set, $\mathcal{E} = [N] \times \chi$ composed of all the cyclic trajectories for any vehicle $u \in \mathcal{U}$, each cloned N times, one for each round: $\mathcal{E} = \{(i, p) : i \in [N], p \in \chi\}$. Such a ground set is partitioned into $N \cdot |\mathcal{U}|$ disjoint subsets $\mathcal{E}_{i,u} = \{(i, p) :\ p \in \chi_u\},\ \forall i \in [N]$ and $u \in \mathcal{U}$.*
- *$\mathcal{I}$ is nonempty collection of subsets of $\mathcal{E}$ defined as $\mathcal{I} = \{S \subseteq \mathcal{E}\ :\ |S \cap \mathcal{E}_{j,u}| \leq 1, \forall (j, u) \in [N] \times \mathcal{U}\}$.*

The pair $M = (\mathcal{E}, \mathcal{I})$ is a matroid (a partition matroid).

Proof. We first establish a one to one mapping between any feasible solution of Problem 3, identified by the variable vectors $\bar{\delta}$ and $\bar{z}$, and a set $S = S(\bar{\delta}, \bar{z}) \in \mathcal{I}$, then we show that M is a matroid.
(One to one mapping.) Given a solution of Problem 3, we can build the corresponding set $S \in \mathcal{I}$ consisting of all the elements (n, p), with $p \in \chi_u$, for which $z_p^u(n) = 1$. Thanks to constraint (c) of Problem 3, there will be at most one element in $S \cap \mathcal{E}_{n,u}$, for each $n \in [N]$ and $u \in \mathcal{U}$. The opposite is also true. If we take any set $S \in \mathcal{I}$, this will have at most one element in each partition subset $\mathcal{E}_{n,u}$ of the matroid M. More specifically, if $S \cap \mathcal{E}_{n,u} = \emptyset$, no trajectory of χ_u is assigned to drone u at round n and $z_p^u(n) = 0$, for all $p \in \chi_u$ and for the selected drone $u \in U$ at round n. If instead the set $S \cap \mathcal{E}_{n,u}$ is not empty, by the definition of the partition matroid M, S will contain only one element (n, p^*) with $p^* \in \chi_u$. This corresponds to assigning $z_{p^*}^u(n) = 1$, and $z_p^u(n) = 0,\ \forall p \neq p^*$. The variables $\bar{\delta}$ are defined according to the setting of $\bar{z}$ in agreement with constraints (b1-b2) of Problem 3, or equivalently using the definition introduced in Observation 4.1.

1. A more complex algorithm with the best approximation known so far, $(1 - 1/e)$, was proposed by Calinescu et al. in [36].

(M is a matroid.) In order to prove that the pair $M = (\mathcal{E}, \mathcal{I})$ is actually a matroid, we must verify the two conditions of Definition 5.1 on M. For condition (1), let us consider A and B, with $A \subset B \subseteq \mathcal{E}$. A and B are sets of indexed cyclic trajectories, for selected vehicles and round indexes. As a consequence, if B belongs to the solution space $\mathcal{I}$ then A also belongs to $\mathcal{I}$ (A represents a solution with fewer trajectories than in solution B). For condition (2), given any A and B in $\mathcal{I}$, such that $|B| > |A|$, we can write $|B| = \sum_{(i,u)\in[N]\times U} |B \cap \mathcal{E}_{i,u}|$ and $|A| = \sum_{(i,u)\in[N]\times U} |A \cap \mathcal{E}_{i,u}|$, because the sets $\mathcal{E}_{i,u}$ constitute a partition of $\mathcal{E}$.

We can see that there exists at least an element $x = (i', u') \in [N]\times\mathcal{U}$ such that $|B\cap\mathcal{E}_{i',u'}| = 1$ and $|A\cap\mathcal{E}_{i',u'}| = 0$. Therefore the element x that satisfies the second condition of Definition 5.1 is such that $\{x\} = B\cap\mathcal{E}_{i',u'}$. Indeed, $x \in B\setminus A$ and $A \cup \{x\}$ is still a solution in $\mathcal{I}$, i.e., $A \cup \{x\} \in \mathcal{I}$. It is straightforward to verify that the definition of the matroid M is consistent with Definition 5.2, hence M is a partition matroid. □

In the following, with p we interchangeably denote either the cycle or the set of nodes traversed by the cycle.

Lemma 5.2. *Problem 3 can be cast as a set function optimization on the partition matroid M defined in Lemma 5.1, where the objective function is*

$$\tilde{f}(S) \triangleq \sum_{n=1}^{N} w(n) \cdot |\cup_{(n,p)\in S} p \setminus \cup_{(i,p)\in S: i<n} p|. \quad (9)$$

Proof. We consider the objective function of Problem 3 and express it in terms of the elements of matroid M. We recall that the objective function of Problem 3 is $\mathcal{W}(N) = \sum_{n=1}^{N} w(n)\cdot\delta(n)$. According to Observation 4.1, $\delta_i(n) = 1$ if $n = \tau_i(N)$ and $\delta_i(n) = 0$ otherwise, for $n \in [N]$. Therefore the value of $\delta(n)$ is the cardinality of the set of targets visited at round n for the first time. Consequently, in terms of the matroid M we can write

$$\delta(n) = |\cup_{(n,p)\in S} p \setminus \cup_{(i,p)\in S: i<n} p|. \quad (10)$$

It follows that $\tilde{f}(S) = \sum_{n=1}^{N} w(n)\cdot\delta(n) = \mathcal{W}(N)$. □

Lemmas 5.1 and 5.2 show that the WPC problem can be cast as the optimization of a set function over a matroid constraint. In the following we prove that this set function is monotone submodular.

Theorem 5.3. *Function $\tilde{f} : \mathcal{I} \to \mathbb{R}$ is a monotone submodular function on the matroid $M = (\mathcal{E}, \mathcal{I})$.*

Proof. We discuss monotonicity and submodularity separately.

Monotonicity proof: Let us consider $A, B \in \mathcal{I}$ such that $A \subset B$. We must show that $\tilde{f}(B) \geq \tilde{f}(A)$.

Given a solution S we define with $\tilde{\tau}_S(i)$ the value of

$$\tilde{\tau}_S(i) \triangleq \min\{n \in [N] : \exists (n,p) \in S \text{ with } p \in \chi, i \in p\},$$

if i is covered, and $\tilde{\tau}_S(i) = N + 1$ otherwise. Notice that, given Lemma 5.2, $\tilde{f}(S)$ can be rewritten as

$$\tilde{f}(S) = \sum_{i\in\Psi} w(\tilde{\tau}_S(i)) \quad (11)$$

Since $A \subset B$, it holds $\tilde{\tau}_A(i) \geq \tilde{\tau}_B(i),\ \forall i \in \Psi$. As a consequence $w(\tilde{\tau}_A(i)) \leq w(\tilde{\tau}_B(i)),\ \forall i \in \Psi$, because $w(n)$ is a non-increasing function of n. Therefore, applying Equation (11), we obtain

$$\tilde{f}(B) = \sum_{i\in\Psi} w(\tilde{\tau}_B(i)) \geq \sum_{i\in\Psi} w(\tilde{\tau}_A(i)) = \tilde{f}(A).$$

Submodularity proof:
To prove the submodularity of $\tilde{f}$, according to Definition 5.3 we must prove that $\forall A \subset B \subseteq \mathcal{E}$ and $e \in \mathcal{E}\setminus B$, $\tilde{f}(A\cup\{e\}) - \tilde{f}(A) \geq \tilde{f}(B\cup\{e\}) - \tilde{f}(B)$.

Let $e = (n_e, p_e)$ be the generic element of $\mathcal{E}\setminus B$. For a generic solution S, applying Equation (11), we obtain

$$\tilde{f}(S\cup\{e\}) - \tilde{f}(S) = \sum_{i\in p_e} \max\{w(n_e) - w(\tilde{\tau}_S(i));\ 0\}.$$

As we observed for monotonicity, $A \subset B$ implies $\tilde{\tau}_A(i) \geq \tilde{\tau}_B(i)$, and consequently $w(\tilde{\tau}_A(i)) \leq w(\tilde{\tau}_B(i)),\ \forall i \in \Psi$, as we recall that $w(n)$ is non-increasing in n. Therefore $\tilde{f}(A\cup\{e\}) - \tilde{f}(A) = \sum_{i\in p_e} \max\{w(n_e) - w(\tilde{\tau}_A(i)), 0\} \geq \sum_{i\in p_e} \max\{w(n_e) - w(\tilde{\tau}_B(i)), 0\} = \tilde{f}(B\cup\{e\}) - \tilde{f}(B)$, which proves submodularity.

□

As a consequence of this observation we can apply Theorem 5.2 to define a greedy approach with $1/2$-approximation of the optimal.

5.2 Algorithm Greedy and Prune (GaP) for weighted progressive coverage

In agreement with the discussion of Section 5.1 on the constant factor approximation of the greedy approach, we propose the adoption of an enhanced greedy algorithm, called Greedy and Prune (GaP) described in Algorithm 1. This algorithm provides a preliminary greedy trajectory assignment phase, followed by a pruning step which removes redundant target points from the selected trajectories, without affecting WPC .

Algorithm 1: Greedy and Prune (GaP)

Input: A set of drones $\mathcal{U}$, a family of candidate cyclic trajectories $\{\chi_u : u \in \mathcal{U}\}$, a number of rounds N, and a weight function $w : [N] \to \mathbb{R}$
Output: An assignment of $|\mathcal{U}|$ ordered lists of cycles $\bar{\mathcal{L}}$, where $\mathcal{L}_u = (p_1^u, \ldots, p_N^u) \in \chi_u$ to each drone $u \in \mathcal{U}$

$\mathcal{R}_{\texttt{assigned}} = (r_1, r_2, \ldots, r_{|\mathcal{U}|}) \leftarrow (1, 1, \ldots, 1)$
$\mathcal{V} \leftarrow \emptyset,\ \bar{\mathcal{L}} \leftarrow \emptyset^{|U|}$
while $\exists u \in \mathcal{U}\ s.t.\ r_u \leq N \wedge \cup_{p\in\chi_u} p \setminus \mathcal{V} \neq \emptyset$ **do**
  $(u^*,\ p^*) = \arg\max_{u\in\mathcal{U}, p\in\chi_u:\ r_u\leq N} w(r_u)\cdot|p\setminus\mathcal{V}|$
  $\chi_{u^*} = \chi_{u^*} \setminus p^*$
  $r_{u^*} = r_{u^*} + 1$
  $\mathcal{V} = \mathcal{V} \cup \{p^* \cap \Psi\}$
  append p_{u^*} to $\mathcal{L}_{u^*}$
prune of redundantly covered targets
return $\bar{\mathcal{L}}$

GaP iteratively builds an assignment of paths for the drones, along the available rounds, by selecting at each step the next tour that maximizes the WPC . When the set of feasible paths χ is restricted, the solution may contain overlapping trajectories, namely trajectories assigned to distinct drones or to the same drone at distinct rounds, containing at least a common target point. In such cases, the pruning algorithm removes this redundancy without affecting WPC. In particular, it removes a target t from a path p if exist an

earlier round with a path p' s.t. $t \in p'$. If overlaps occur in the same round, the algorithm prune every occurrence but one. Clearly, as the algorithm removes only redundant targets and leaves them in earliest rounds, it provides the same value of WPC as a pure greedy approach. Therefore we derive the following corollary.

Corollary 5.1. *GaP achieves* $1/2$*-approximation of the optimal solution to the* WPC maximization problem *under restricted sets of trajectories*[2].

That is, the WPC obtained by the trajectories selected by GaP is at least half of the maximum that can be obtained by optimally choosing the drone trajectories within the restricted feasible sets χ_u.

We recall that the formulation of weighted progressive coverage generalizes the other coverage metrics described in Section 3.1. The total coverage metric of Equation (3) and the accumulative coverage metric of Equation (5) can be obtained by setting the weight parameters $w(k)$, $k = 1, \ldots, N$, to specific values, such that $w(k)$ is non increasing with k. Therefore Theorem 5.3 is still valid for assessing monotonicity and submodularity of the total and of the accumulative coverage metric, which implies that the Greedy and Prune approach of Algorithm 1 provides a constant factor approximation of $1/2$ also for these metrics. The result naturally extends to the case of unrestricted sets of trajectories, considering $\mathcal{C}_u = \chi_u$.

Theorem 5.4. *The complexity of GaP is* $\mathcal{O}(|\mathcal{U}| \cdot |\mathcal{X}| \cdot |\Psi| \cdot N)$

Proof. The greedy assignment of trajectories runs in $\mathcal{O}(|\mathcal{U}| \cdot |\mathcal{X}| \cdot |\Psi| \cdot N)$. In fact, it iterates in the while loop (lines 3-8) at most $N \cdot |\mathcal{U}|$ times, because at each iteration at the algorithm selects a new path for a drone u, and the related drone index r_u is increased. Its body loop can easily be implemented to run in $\mathcal{O}(|\mathcal{X}| \cdot |\Psi|)$, which is the time of dominant instruction at line 4. In fact, assuming that the weighted function $w(x)$ is constant ($\mathcal{O}(1)$), the $\arg\max$ function runs in $\mathcal{O}(|\mathcal{X}| \cdot |\Psi|)$ while the other instructions in $\mathcal{O}(|\Psi|)$.

The pruning step can be implemented in polynomial time in $\mathcal{O}(|\mathcal{U}| \cdot |\Psi| \cdot N)$. In fact, the pruning step iterates over the trajectories selected in the while loop, and updates a list of visited targets, sorted by round number, so that it can remove redundant targets already in the list, and shortcut the related trajectories. □

5.3 Generation of the restricted sets of paths

For the formulation of Problem 3, we have assumed that a set of feasible drone trajectories is part of the definition of the WPC optimization problem instance. In the following we discuss a technique to compute a set of efficient candidate trajectories, taking account of the energy limitations of the drones. Algorithm 2 is used to generate a subset of feasible trajectories for our simulations in Section 6 and prototype experiments in Section 7.

The algorithm initially calculates an approximate solution of TSP over the set of targets points $\Psi \cup \{d_u\}$, for each drone $u \in \mathcal{U}$. This step is based on the 1.5-approximation algorithm for TSP proposed by Christofides [38] which produces a route τ_u. The target points are then numbered according to their order of visit in τ_u, starting from the depot: $\tau_u =< d_u, v_1, v_2, \ldots, v_{|\Psi|} >$. The algorithm computes the restricted set of trajectories for drone u, by extracting all the possible sub-tours from τ_u which, connected to the depot d_u, meet the energy limitation b_u, i.e. all the sub-tours τ'_u for which $\omega_u(\tau'_u) \leq b_u$, where $\omega_u : \mathcal{C}_u \to \mathbb{R}$ gives the energy consumption of traversing τ_u according to the energy model of drone u.

As the drones may have different depots, energy availability and energy consumption models, the algorithm iterates on each of them to produce different sets of trajectories χ_u, $\forall u \in \mathcal{U}$, generating at most $|\Psi| \cdot (|\Psi| - 1)/2$ candidate trajectories for each drone. If the energy model allows polynomial time verification of the feasibility of a trajectory, Algorithm 2 is also polynomial in the number of drones $|\mathcal{U}|$ and targets $|\Psi|$.

Algorithm 2: Drone-trajectory generation

```
1  input : drones U, targets Ψ, energy consumption
           models ω_u : C_u, ∀u ∈ U → ℝ
   output: Candidate trajectories {χ_u : u ∈ U}
2  sol= ∅;
3  for u ∈ U do
4  |  χ_u = ∅;
5  |  τ_u = 1.5-TSP(Ψ ∪ {d_u});
6  |  sort(τ_u); //τ_u =< d_u, v_1, v_2, ..., v_|Ψ| >;
7  |  for i ∈ {1, ..., |Ψ| − 1} do
8  |  |  for j ∈ {i + 1, ..., |Ψ|} do
9  |  |  |  τ'_u =< d_u, v_i, ..., v_j, d_u >;
10 |  |  |  if ω_u(τ'_u) ≤ b_u then
11 |  |  |  |  χ_u = χ_u ∪ {τ'_u};
12 |  sol = sol ∪ {χ_u};
13 return sol
```

6 Performance evaluation

In the following we give a simulative evaluation of the algorithms discussed in this paper, while we devote Section 7 to real prototype experiments. We recall that both Problem 3 and the GaP algorithm aim at optimizing weighted progressive coverage, in its general formulation given in Equation (2), for any setting of the weight function $w(k)$, $k = 1, \ldots, N$. In this section we consider the two practical coverage metrics introduced in Section 3.1, namely *total coverage* $\Delta(N)$ defined by Equation (3), and *accumulative coverage* $\mathcal{A}(N)$ defined by Equation (5). Total and accumulative coverage are instances of WPC obtained by setting $w(k) = 1$ for total coverage in N rounds, and $w(k) = N - k + 1$ for accumulative coverage in N rounds, $\forall k = 1, \ldots, N$.

We hereby refer to the two variants of the optimal solution with the names TC-OPT and AC-OPT, for total coverage and accumulative coverage optimization, respectively. Similarly, the GaP algorithm variants corresponding to the two objectives are called TC-GaP and AC-GaP.

In the following, we compare the aforementioned approaches against a previous proposal, introduced by Kim

2. A relaxation to the continuous and consequent random rounding of the problem 3 conducted with the technique described in [36] guarantees an approximation factor of $1 - 1/e$.

et al. [11], hereby referred to with the name Tree Cover Trajectory Planning Algorithm (TCTPA) as it is based on the creation of tree covers rooted at the depot locations. In Section 6.2 we discuss this previous approach, explaining its limitations when applied to the scenario considered in this paper, and showing the way in which they are addressed.

6.1 Energy model and restricted set of paths

While our algorithms work independently of the specific technique adopted to generate the restricted set of paths χ_u for each drone, in this experimental section we adopt the drone trajectories generated by Algorithm 2, considering that the energy consumed by a drone is proportional to the traversed distance and length of hovering time.

Concerning hovering, if ϕ_i is the necessary time to inspect the target point i, the energy consumption related to target inspection is assumed to be proportional to ϕ_i. We also assume $\phi_{d_u} = 0$ for each depot d_u of any drone $u \in \mathcal{U}$.

Given any two points $i, j \in \Psi$ we denote with ℓ_{ij} the distance that a drone needs to traverse to move from point i to point j. With an abuse of notation we say that $i \in p$ if target point i is traversed by p and that $(i, j) \in p$ if the target points i, j are traversed by p in a sequence.

The energy expenditure for traversing a path p, including hovering on the traversed target points, is $E(p) \triangleq a \cdot \sum_{i \in p} \phi_i + b \cdot \sum_{(i,j) \in p} \ell_{ij}$, where a and b are dimensional coefficients which reflect the energy consumption in energy units (eu) for a second of inspection of a target and a meter of flight, respectively. In all the experiments we consider $a = 1$eu/sec and $b = 1$eu/m.

6.2 Tree Cover Trajectory Planning Algorithm (TCTPA)

In [11] the authors address the problem of planning multiple drone trajectories to inspect a number of target points positioned in an area of interest. The primary goal is to ensure complete coverage of the target points, while minimizing the mission completion time, i.e. the time at which the last drone returns to its depot.

TCTPA builds a graph whose vertices coincide with the target points and depots. Then, it builds a set of $|\mathcal{U}|$ balanced tree covers rooted at the depot stations, and convert each tree to a drone trajectory by means of a technique inspired to the Christofides algorithm for the TSP [38].

The authors consider a multiple depot scenario, but assume each device has unlimited energy, therefore the drone squad is always able to inspect all the targets in a unique round. Unlike this previous approach, our algorithms work in a more general scenario, where the flight autonomy of each vehicle is constrained because of the limited energy availability of each drone. To have fair comparisons, we made TCTPA work under limited energy availability, letting drones fly in multiple rounds N, possibly depleting their batteries before the completion of the target inspection mission. To make TCTPA produce N tours for each drone, we let it work as if the number of available drones was $N \cdot |\mathcal{U}|$, considering N virtual clones for each physical drone, each with its logically replicated depot, and we let each physical drone traverse one of its clones' tours at each round.

Figures 3, 4, and 5, show the execution of AC-OPT, AC-GaP, and TCTPA, respectively, in a field of interest of 600m $\times 600$m. Three drones are launched from different depots located on the bottom border of the field, and are required to inspect 50 target points randomly generated in the field area. Each drone speed is 8m/s, the drone battery is set to allow a continuous flight of about 2000m in a single round. The inspection of each target requires 5 sec of hovering time.

The figures show that AC-OPT and AC-GaP definitely succeed in designing trajectories which, under the battery limitation, contain more target points in the early rounds. The first round trajectories produced by AC-OPT and AC-GaP for the three drones are almost the same and contain a number of targets which is much higher than the average number of targets in the successive rounds. Both these algorithms complete the mission of inspecting the 50 targets in just two rounds. By contrast, TCTPA produces a more uniform distribution of targets in the exploration time, and ends up requiring one additional round for the two drones on the left of the figure.

These figures show that, in the addressed scenario, TCTPA performs worse than AC-OPT and AC-GaP in terms of accumulative coverage as it does not give priority to target coverage in the early rounds. They also show that TCTPA requires a longer completion time (in number of rounds) than the other two algorithms, which is its specific objective. Notice that in emergency critical settings, the time interval between consecutive rounds is devoted to recharging, maintenance and data offloading, which are time consuming activities. Hence a solution that minimizes the number of rounds, together with early coverage, such as AC-GaP, is preferable in these settings.

6.3 Performance comparisons

In this section we give performance comparisons through simulations, evaluating several key performance metrics under different settings. Where not otherwise stated, the experiments consider a drone fleet of 5 vehicles, monitoring a squared area of interest of 2000m $\times 2000$m with randomly located target points requiring 5sec of hovering time, with depots uniformly deployed at 100m of distance, out of the area of interest, to reflect the inaccessibility of the monitored field. Each of the drones executes a maximum number of $N = 20$ rounds of flight, with a speed of 8m/sec, under a uniform battery constraint b for each drone, which allows a flight of 7200m at constant speed (for 15min). Each point in the plots is the result of 30 runs, and the error bars denote one standard deviation of uncertainty.

6.3.1 Progressive coverage percentage

In the experiments we will evaluate the algorithms in terms of several performance metrics including the achieved *progressive total coverage percentage*, hereby defined as the percentage of target points covered up to a given round (according to Equation (3)), evaluated in progressive rounds. A progressive coverage percentage of $x\%$ at round n reflects a situation in which $x\%$ of the target points have been covered in any round from the first to the n-th.

Fig. 6 compares the two optimal and the two greedy approaches with TCTPA, in a setting with 225 target points, in terms of progressive total coverage achieved round by round. It is worth noting that the area under the plot until

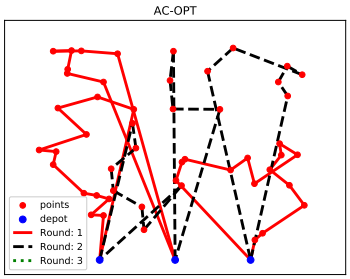


Fig. 3. Consecutive flights under AC-OPT.

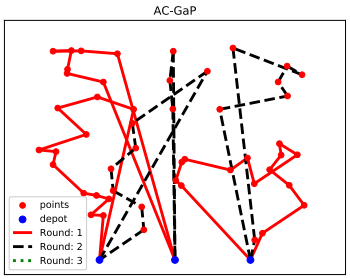


Fig. 4. Consecutive flights under AC-GaP.

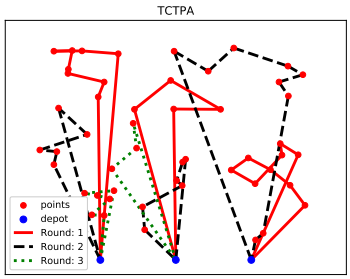


Fig. 5. Consecutive flights under TCTPA.

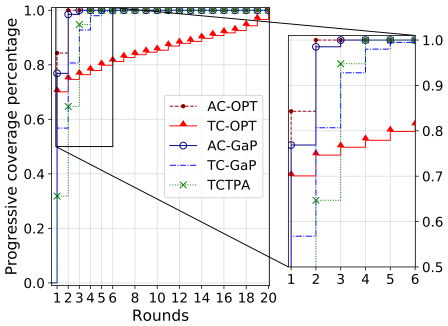


Fig. 6. Progressive coverage (225 target points).

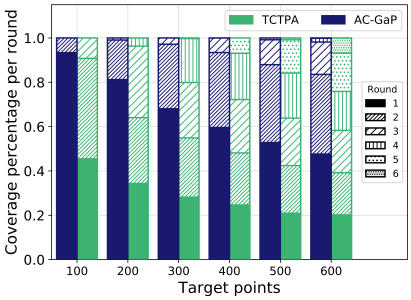


Fig. 7. Coverage percentage (varying nr. of targets).

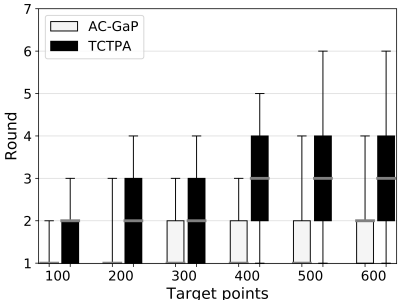


Fig. 8. Quartiles of progressive coverage.

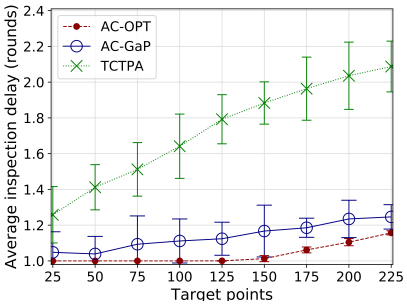


Fig. 9. Average inspection delay in rounds.

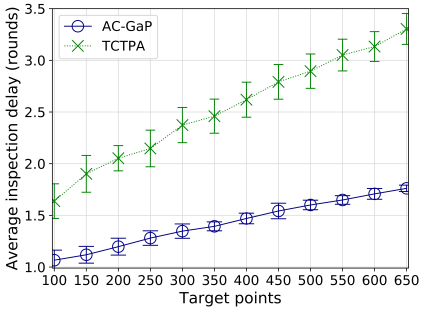


Fig. 10. Average inspection delay in rounds.

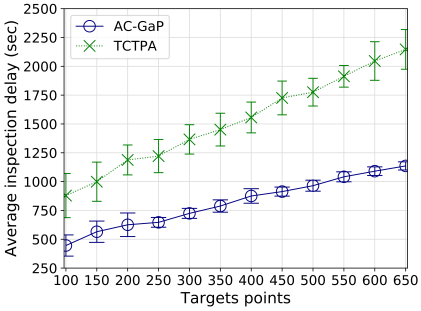


Fig. 11. Average inspection delay in seconds.

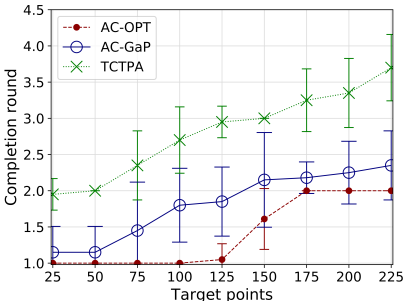


Fig. 12. Completion round.

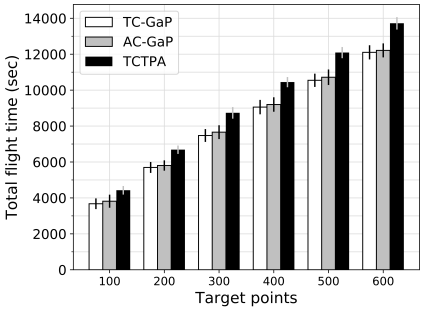


Fig. 13. Flight time until mission completion.

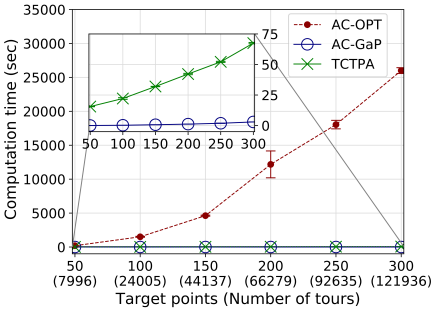


Fig. 14. Algorithm computation time.

round n is equal to the accumulative coverage at round n divided by the total number of target points. The figure shows that, after AC-OPT, the algorithm that performs the best, both in total coverage and in accumulative coverage, is AC-GaP, with an excellent approximation of the optimal. Both the algorithms achieve very high coverage in the early rounds of execution because they adopt decreasing weights $w(n)$ as prescribed by Equation 2 to give decreasing priority to the coverage achieved at any round $n = 1, \ldots, N$.

It is interesting to notice that in terms of progressive total coverage, TC-GaP performs better than TC-OPT, until round 20 in which both algorithms achieve 100% of total coverage. This is due to the fact that by setting the number of rounds to $N = 20$, and the weight parameters to $w(n) = 1$, TC-GaP and TC-OPT make any of the 20 rounds as important as the others in terms of coverage. None of these algorithms aim at obtaining high coverage in the early rounds, but only at the 20-th round. An algorithm that obtains zero coverage at the first 19 rounds and obtains complete coverage at round

20, has the same value of total coverage $\Delta(N = 20)$ as another algorithm that obtains complete coverage since the very first round. Finally, we observe that TCTPA has a very good performance in this setting in terms of total coverage after round 3. Nevertheless in the early rounds TCTPA performs poorly with respect to all the other algorithms, and in particular, in the first round it achieves only less than half the coverage obtained by our AC-GaP.

Figure 7 shows the coverage percentage achieved at any round by AC-GaP and TCTPA in settings with a number of target points growing from 100 to 650. The stacked bars show the percentage of points covered at each round (i.e., from the points covered at the first round, corresponding to the stack filled with solid texture, up to the last used round). For example, when the number of target points is 100, AC-GaP covers more than 90% of the targets within the first round (solid color block), and the remaining targets in the second round (block with dense diagonal lines). In contrast, AC-GaP covers only a little more than the 40% of the targets within the first round, and a little more than the 90% by the end of the second round, although it also needs a third round to complete target coverage. Due to the large size of the problem instances, we do not include the optimal policies in this figure, nor do we include the TC-GaP heuristics because its purpose is not to obtain high coverage in the early rounds but only within a given round N. The figure shows that also in this setting, for any round $n = 1, \ldots, N$ the coverage obtained by AC-GaP is always higher than with TCTPA. These results match the analysis of Figure 8 where we use a box-with-whiskers plot to show, by varying the number of target points, the round at which the algorithms achieve the different coverage quartiles. The figure highlights that AC-GaP meets all the quartiles in an earlier round than TCTPA, demonstrating the superiority of AC-GaP in total coverage and in accumulative coverage per round.

6.3.2 Average inspection delay

The average inspection delay, in number of rounds, has been introduced with Equation 7. With Theorem 3.1 we proved that AC-OPT is optimal both for accumulative coverage and for average inspection delay in rounds. Figures 9 compares our algorithms in terms of this metric, for a growing number of target points and confirms the optimality of AC-OPT. Due to the high computational time of the optimal solution (discussed in Section 6.3.4), we only compare the optimal AC-OPT with the heuristic algorithms AC-GaP and TCTPA for small problem instances in Figure 9, whereas when the size of the problem instance grows, as shown in Figure 10, we compare the two heuristics AC-GaP and TCTPA alone. As explained by Theorem 3.1, AC-OPT performs always better than the other algorithms in terms of average inspection delay in rounds. Indeed, Figures 9 and 10 show that the inspection delay in rounds of algorithm TCTPA diverges from the one of AC-OPT and AC-GaP, showing that the latter algorithms excel in achieving early monitoring of target points. Figure 11 shows a different experiment in which we still consider the average inspection delay, but we measure it in seconds, by considering zero waiting time between two consecutive rounds, in order not to penalize TCTPA with this setting. The figure shows that also in these terms AC-GaP performs better than TCTPA, due to the fact that TCTPA prioritizes the completion time of the algorithm, which is the inspection time of the last target point, at the expense of an increased inspection delay of the other targets. Despite the fact that mission completion time is the specific objective of TCTPA, Figure 12 shows that for the considered scenario, TCTPA always completes the mission with almost twice the number of rounds used by AC-GaP.

6.3.3 Energy cost

Figure 13 shows the total flight time over all the 20 rounds, considering all 5 drones, for the three heuristics TC-GaP, AC-GaP and TCTPA. Notice that since we use the linear energy model discussed in Section 6.1, this measure directly reflects the energy expenditure of drones.

The graph shows that the difference between the algorithms in terms of total energy expenditure is negligible, although in small favor of the greedy approaches.

This explains that the superiority of the greedy approaches resides on the way drone visits are scheduled along the rounds and within single rounds, rather than in the minimization of the traversed paths.

6.3.4 Computation time

In Figure 14 we compare the algorithms in terms of computation time. For this and for all the previous experiments we adopted a homemade simulator, programmed in Python, and we ran the ILP optimization problems using the Gurobi optimizer [39]. We ran all the experiments on a Lenovo X3550 M5, with 2 CPUs Intel(R) XEON(R) E5-2650 @ 2.20GHz with 16 cores and 32 GB RAM [40].

In order to stress the optimal algorithm while keeping the problem size small enough to be executed in a reasonable time to have enough runs for a reliable experimental evaluation, we designed a new setting with 10 drones, and a growing number of target points, from 50 to 300. This experiment shows that the computation time of the optimal solution grows prohibitively high also for this small a setting. With 300 target points the computation time is already in the order of 7 hours. Such a high processing time makes the optimal solution inapplicable to emergency critical applications which require prompt intervention of the aerial network and do not tolerate delays. The figure also contains a zoomed plot which shows the difference between TCTPA and AC-GaP in terms of running time. Though negligible, this performance parameter is also in favor of AC-GaP.

7 Real field experiments

In this section we experimentally investigate the applicability of our proposal and validate simulation results. The equipment we use for the test-bed includes a fleet of drones — able to follow way-point missions using GPS coordinates — and, a laptop — needed to run the algorithms, compute the trajectories and to provide them to drones.

First, we discuss the major challenges we dealt with while performing the mission. Then, we compare the simulation results with real-field experiments, under the same

settings. Finally, we compare our algorithms against state-of-art solutions. We evaluate two scenarios, with battery replacement and with battery recharging between consecutive rounds. To conclude the analysis, we also use our approach to address area coverage, i.e., a different objective function for which the algorithm is not specifically designed.

7.1 Real-field Scenario

In the experiments we simulate a rescue mission with several survivors needing help. The experiments are performed in a university soccer field in Amman, Jordan, with an area of interest of around $140m \times 90m$, with $|\Psi| = 40$ randomly distributed target points, modeling survivors. For each experiment, we perform 3 different runs. Where not otherwise stated, we use a fleet of four drones, with four different depots, located at the borders of the field, 5 meters apart from each other. The drones are DJI Flame Wheel 550 (F550)[25] hex-rotor combined with a Naza-M V2 [41] flight control system, as showed in Figure 15. The Naza-M V2, through the GPS module, allows drone localization and enable way-point missions. All the drones were equipped with LIPO batteries, 3500 mAh, 11.1 v, 25 C which allow around 7 minutes of consecutive flight with a speed under 5m/s. The time required for a full recharge is around 1 hours. Notice that, temperature can affect the LIPO battery's performance. Most of the experiments were carried out in February, with a temperature from 7°C to 15°C.

We consider a maximum number of rounds $N = 7$. After the end of each round, we recharge or replace the drone batteries and measure the required times. These times are then used to evaluate two scenarios in which drone batteries are: a) replaced with fully-charged batteries, or b) recharged, between consecutive rounds.

Table 2 lists all the relevant experimental parameters.

Experiment workflow. To reproduce a real application scenario, we take the following steps:

- We perform a preliminary computation of the set of candidate trajectories for each drone. We compute these trajectories by using the Algorithm 2, which is also adopted in the simulations.
- We run the algorithms AC-OPT, AC-GaP and TCTPA to obtain the selected trajectories and their schedule in rounds.
- We let the drones fly along the selected trajectories, taking measures at each depot and target point.

Fig. 15. DJI F550 Hex-rotor with Naza-M V2

TABLE 2
Experiment settings

Field	Jordan University Stadium, Amman, Jordan
Local time	9.00-17.00 a.m.
Local temperature	$+7 - 15°C$
Wind Speed	1.0 to 2.1 m/s
Field Size	140 x 90 (meters)
Number of drones	4
Number of targets	40
Hovering time	90 seconds
Height above the ground	4 to 10 meters
Max number of rounds	7

Moreover, as experiments are conducted on a flat soccer field, we consider flight trajectories such that drones vertically take off, fly each at a different constant height, and vertically land to the depot.

To precompute candidate trajectories we assume that drones fly at constant speed when moving from target to target, and hover for about 90s above each inspected target. To take into account the different speed that the drones may reach during flights of different lengths, we consider a uniform speed of 1.3 m/s for flight distances of up to 30 meters, 2m/s up to 50 meters, and 2.5m/s for longer distances.

We model the drone energy availability on the basis of the information advertised by the producer and our previous experience: around 450 seconds of flight-time when the speed is up to 5m/s. We adopt this value by considering that in similar experiments the drones never exceeded 5m/s of speed. It is worth to notice that these values affect the set of feasible trajectories in any algorithm that takes energy limitation into account. Nevertheless, the errors in this preliminary phase have a negligible impact in the algorithm comparative evaluation, with inspection delays that are just few seconds away from the offline calculated estimate.

7.2 Mission issues and challenges

We now discuss the major challenges we faced while deploying drones and performing missions.

Aerial collisions. To avoid potential collisions between drones, we used different flight heights for each drone. We underline that thanks to the pruning algorithm phase (5.2), two drones never visit the same target, which reduces collision risks considerably. In the experiments for which multiple drones shared the same depot, take off and landing procedures must be performed in a serial schedule. Moreover, drones can collide with physical objects. In this case two solutions are possible. First, drones may have integrated obstacle avoidance systems [24]. Second, drones can integrate an online collision avoidance mechanisms in the UAV control pipeline, to dynamically change the trajectories. For example, the work in [42], presents an online avoidance maneuver that alters the reference trajectory of the vehicle by adding an offset in height in case of a possible collision.

In the proposed presented in this section, we did not have any physical obstacle along the field, and each drone had its own depot.

Errors in the candidate trajectory pre-computation. As drones have limited energy, estimating their flight capa-

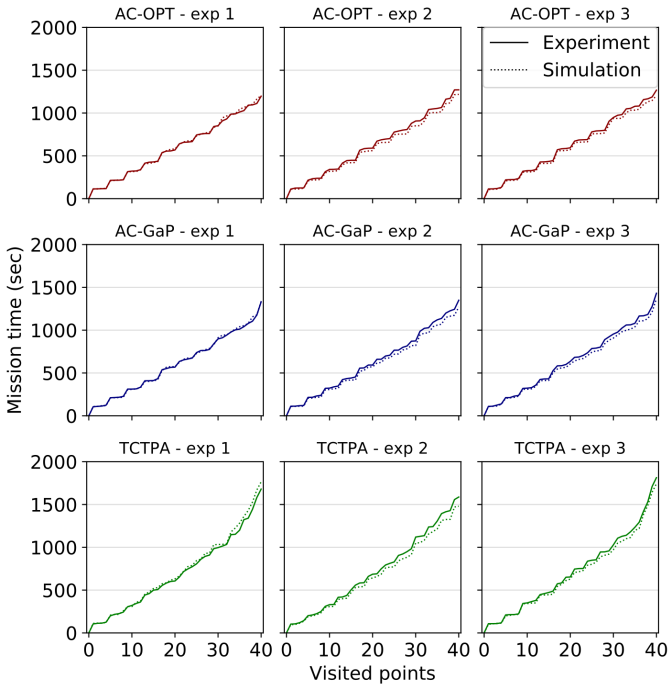


Fig. 16. Simulation vs Experiment visit times.

bilities is a significant challenge for a successful mission. If the algorithms underestimate the energy consumption of drones, they may produce unfeasible paths, leading to emergency landings. As drones manufacturers give little information on the energy consumption model — they typically describe the device energy availability in terms of indoor hovering time or flight at uniform speed — we measured the drones' capabilities by analyzing real field past experiments. Specifically, when computing the set of candidate trajectories χ_u for each drone u, we estimate the flight capabilities by averaging and interpolating measures obtained in similar environmental conditions, considering the different activities such as take-off, flight at uniform speed, hovering and landing. Then, we slightly reduced the value of this estimate to have a conservative evaluation of the drone capabilities to make the solution more tolerant to unpredictable conditions, such as adverse wind.

7.3 Validation of simulation results

We now compare simulations and experiments conducted on the real test-bed using the same settings described in Table 2. We do not show a comparison between simulated and real experiments in terms of rounds as they produce the same results. Figure 16 shows the time elapsed (y-axis) until the visit of an increasing number of target points (x-axis) for all the compared algorithms. For this experiment, we considered zero waiting time between two consecutive rounds. The dotted lines represent the offline simulated trajectories, while the solid ones are the real field experiments. The lines show only very small differences between simulations and experiments. Confirming this analysis, Figure 17 evaluates the difference between simulations and experiments in terms of the Mean Absolute Percentage Error (MAPE). The figure shows that the MAPE of the average inspection delay is always below 7%. This slight discrepancy is mostly due to uncontrollable environmental factors (e.g., wind), measurement errors, and drone acceleration/speed in the real field.

Thus, our first set of experiments validates simulation results.

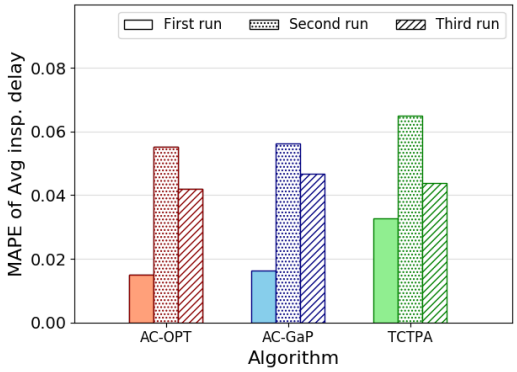


Fig. 17. Simulation vs Experiment average inspection delay in seconds.

7.4 Progressive coverage evaluation

We experimentally compare our algorithms with previous solutions. We evaluate two scenarios, which differ in the operation between two consecutive rounds: the first implements battery replacement while the second provides battery recharging. Obviously, battery recharging has a significant impact on overall mission time.

Figure 18 shows the progressive coverage percentage for the three algorithms (see section 6.3.1), with respect to mission rounds. Specifically, AC-GaP performs very well, close to AC-OPT. Both algorithms achieve very high coverage in the early rounds of execution, with complete coverage before round 3 for AC-OPT and round 4 for AC-GaP. Instead, TCTPA performs poorly, as it requires 7 rounds to complete the visit.

We further investigate progressive coverage in the two scenarios in terms of elapsed time from the mission start.

In the battery replacement scenario, shown in Figure 19, AC-OPT and AC-GaP perform better than TCTPA, reducing the average inspection delay of about 25% (see Table 3). This gap heavily increases in proportion to the time spent at the depot for battery replacement and device maintenance, due to the higher number of rounds required by TCTPA.

If we consider battery recharging between rounds, as we show in Figure 20, the performance of TCTPA degrades significantly with respect to AC-OPT and AC-GaP, which however present a step-wise increase in progressive coverage percentage. TCTPA takes 5000s to cover about 50% of targets, while at the same time, AC-OPT and AC-GaP cover 75% and 72% of targets, respectively.

TABLE 3
Average Inspection Delay on real-field experiments

Average Inspection Delay	AC-OPT	AC-GaP	TCTPA
# of rounds	0.85 ± 0.02	0.9 ± 0.04	2.32 ± 0.13
# of seconds - with battery replacement	727 ± 15	744 ± 21	992 ± 11
# of seconds - with battery recharging	3578 ± 62	3717 ± 127	6147 ± 247

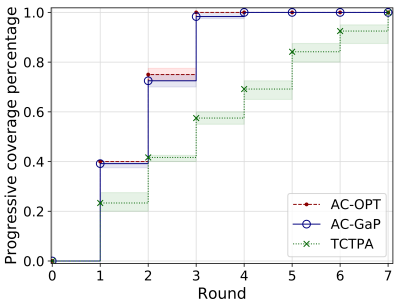


Fig. 18. Progressive Coverage (Rounds)

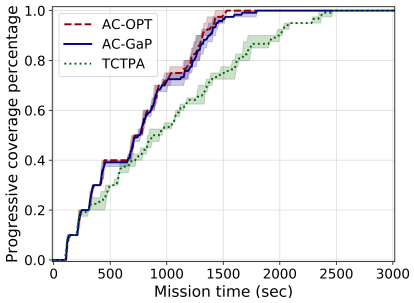


Fig. 19. Progressive Coverage with battery replacement (sec)

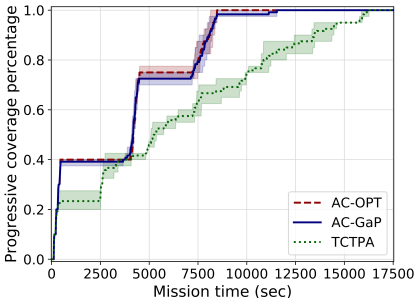


Fig. 20. Progressive Coverage with battery recharging (sec)

Finally, Table 3 shows the average inspection delay (see section 6.3.2) with respect to the number of rounds and time elapsed from the mission start, in both the battery replacement and recharging scenarios. The results clearly show that AC-OPT and AC-GaP are comparable and outperform the TCTPA. In detail, the optimal and greedy algorithms perform better than TCTPA, reducing the average inspection delay in seconds of around 25% and 40%, with battery replacement and recharging, respectively.

To conclude, our second set of experimental results confirm the real field applicability of our approximation algorithm, which outperforms TCTPA. The experiments also reveal that the new WPC metric correctly reflects the algorithm capabilities to provide early target inspection, especially in battery recharging settings.

7.5 Energy model - sensitivity analysis

As we previously discussed when we addressed the challenges of a real field test-bed, errors in the energy model may result in the computation of inefficient or infeasible trajectories due to an incorrect estimate of the drone capabilities. We studied the effects of under-estimating and over-estimating the drone energy consumption considering different error values. Due to the randomness of the target deployment over the area, in none of the experiments we had a situation in which drones had the exact energy availability to complete their trajectories with no residual energy. Therefore the algorithms showed a good flexibility to errors in the energy model. We had to introduce an error as high as the 30% before being able to observe significant differences in number of rounds with respect to the case of a correct energy model.

In a scenario with 20 randomly deployed target points and 4 drones an overestimation error of 30% in the drone energy consumption produces shorter trajectories than necessary, requiring 1 additional round in all the algorithms. In contrast, an underestimation error of 30% produced unfeasible trajectories, so that some drones were not able to complete their tasks and had to return to the depot, requiring additional time to identify the un-visited targets and re-compute a solution to complete the mission.

7.6 Area coverage evaluation

In this section, we investigate the performance of our approach when the application requires coverage of an entire area, and not just of some target points. To extend our target coverage algorithms to the new application scenario, we provide a square tessellation of the area (with tile sized such that they can be inscribed in the drone monitoring range) and set a target at the center of each tile with null hovering time requirement. With this setting, by inspecting each target, a drone entirely covers the area of interest. For this evaluation, we consider an algorithm specifically designed to obtain area coverage: the cooperative *Sweep* algorithm — an extension of the approach proposed in [23] to drone networks. The algorithm partitions the whole area among the available drones, taking into account their capabilities and different initial locations. Each partition is then covered with a rectangular scale-line pattern.

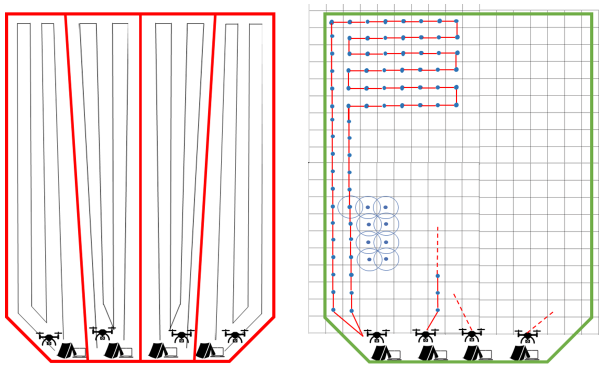
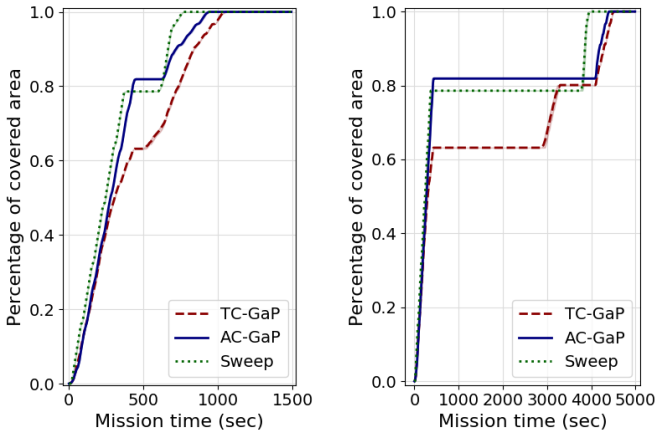


Fig. 21. Cooperative coverage path planning sample scenario.

The drawing of Figure 21 shows an example of the algorithm trajectories in the addressed scenario. The left image represents the *Sweep* algorithm. Notice that, if a path requires to much energy to be executed by a drone in a single trip, we truncate it and we reassign the remaining part to a subsequent round. The right image exemplifies the setting of target points to address this scenario with target coverage algorithms.

We run two different experiments to test the algorithm performance while covering the whole area with 4 drones, both with replacement and recharging actions between consecutive rounds. We consider a 360°-camera with a monitor range of 2.5 meters. We do not consider hovering time, but we slow down the maximum speed to 2m/s, to allow the camera to collect data during the flights.

In this scenario we compare Sweep with our greedy algorithms: AC-Gap, which employs the accumulative cov-



(a) Battery replacement (b) Battery recharge

Fig. 22. Percentage of covered area.

erage metric, and TC-GaP, which employs the total coverage metric. Figure 22(a) and 22(b) show the percentage of covered area since the beginning of the mission, in case of battery replacement and battery recharge, respectively. While all the algorithms have the same performance in number of rounds to achieve total coverage (i.e., they cover the whole area in 2 rounds), AC-Gap covers around 82% in the first round, while Sweep only the 78% and TC-GaP the 62%. In particular, the figures show that AC-GaP visits most of the points at the beginning of the mission, which is extremely helpful in critical situations, especially when the batteries are being recharged. In Figure 22(b) AC-GaP visits the 82% of the points in the first 8 minutes of mission, while Sweep needs more than 1hour to reach the same coverage percentage.

Finally, we note that despite the performance differences shown so far, the three algorithms need about the same time to achieve a complete coverage, for both the battery replacement and battery recharging scenarios. Sweep needs 800s and 4000s, AC-GaP needs 900s and 4350s, and TC-GaP completes the mission in 1050s and 4400s, respectively, in the two scenarios.

8 Conclusions

The work addresses the problem of assigning location based tasks to a fleet of drones in an emergency critical scenario, to ensure early inspection of target locations. We propose a novel metric, called weighted progressive coverage, to measure the capability of a trajectory planning algorithm to provide early target inspection, and formulate a related optimization problem. We show that weighted progressive coverage generalizes traditional metrics of coverage, as well as a new notion of accumulative coverage which is directly related to early coverage capabilities. Due to the NP-hardness and high computation time of the optimal approach, we propose a new polynomial time algorithm with a constant factor approximation of $1/2$ of the optimal. We study the proposed algorithm by means of extensive simulations, showing that it outperforms previous approaches in terms of all the metrics of coverage, average inspection delay, energy consumption and computation time. We also validate our simulation results through prototype experiments and demonstrate the applicability of our approach in real field scenarios.

9 Acknlowledgements

This work was supported by the North Atlantic Treaty Organization (NATO), under the Science for Peace and Security grant, number SPS G4936.

References

[1] N. Strochlic, "Surprising ways drones are saving lives, National Geographic," 2017. [Online]. Available: https://www.nationalgeographic.com/magazine/2017/06/explore-drones-for-good/

[2] M. McNabb, "Drones for good: UNICEF is calling on drone operators for vaccine delivery. 2018. [Online] https://dronelife.com/2018/06/14/drones-for-good-unicef-is-calling-on-drone-operators-for-vaccine-delivery," 2018.

[3] N. Bartolini, A. Coletta, and G. Maselli, "On task assignment for early target inspection in squads of aerial drones," in *39th IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2019.

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

[5] P. Toth and D. Vigo, *Vehicle routing: problems, methods, and applications*. SIAM, 2014.

[6] J. Fakcharoenphol, C. Harrelson, and S. Rao, "The k-traveling repairman problem," in *Proceedings of the 14th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2003.

[7] C. S. Martin and M. R. Salavatipour, "Minimizing latency of capacitated k-tours," *Algorithmica*, vol. 80, no. 8, pp. 2492–2511, Aug 2018. [Online]. Available: https://doi.org/10.1007/s00453-017-0337-x

[8] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, pp. 1–10, 2011.

[9] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter, "On the capacitated vehicle routing problem," *Mathematical programming*, vol. 94, no. 2-3, pp. 343–359, 2003.

[10] P. Oberlin, S. Rathinam, and S. Darbha, "A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem," in *IEEE American Control Conference*, 2009.

[11] 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 Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3156–3166, Nov 2017.

[12] F. Afghah, A. Razi, J. Chakareski, and J. Ashdown, "Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles," *IEEE MiSARN*, 2019.

[13] S. Lee and J. R. Morrison, "Decision support scheduling for maritime search and rescue planning with a system of uavs and fuel service stations," in *IEEE International Conference on Unmanned Aircraft Systems (ICUAS)*, 2015.

[14] Q. Wu, Y. Zeng, and R. Zhang, "Joint trajectory and communication design for multi-uav enabled wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 2109–2121, 2018.

[15] O. Zorlu, "Routing unmanned aerial vehicles as adapting to capacitated vehicle routing problem with genetic algorithms," in *IEEE 7th International Conference on Recent Advances in Space Technologies (RAST)*, 2015.

[16] H. Binol, E. Bulut, K. Akkaya, and I. Guvenc, "Time optimal multi-uav path planning for gathering its data from roadside units," *IEEE VTC-Fall*, 2018.

[17] M. H. Dominguez, S. Nesmachnow, and J.-I. Hernández-Vega, "Planning a drone fleet using artificial intelligence for search and rescue missions," in *IEEE 24th International Conference on Electronics, Electrical Engineering and Computing (INTERCON)*, 2017.

[18] J. Faigl and P. Váňa, "Unsupervised learning for surveillance planning with team of aerial vehicles," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 2017.

[19] S. Islam and A. Razi, "A path planning algorithm for collective monitoring using autonomous drones," in *53rd IEEE Conference on Information Sciences and Systems (CISS)*, 2019.
[20] T. Setter and M. Egerstedt, "Energy-constrained coordination of multi-robot teams," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 4, pp. 1257–1263, 2016.
[21] M.-I. Popescu, H. Rivano, and O. Simonin, "Multi-robot patrolling in wireless sensor networks using bounded cycle coverage," in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2016, pp. 169–176.
[22] A. Yazici, G. Kirlik, O. Parlaktuna, and A. Sipahioglu, "A dynamic path planning approach for multirobot sensor-based coverage considering energy constraints," *IEEE Transactions on Cybernetics*, vol. 44, no. 3, pp. 305–314, 2013.
[23] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Transactions on robotics*, vol. 22, no. 3, pp. 507–522, 2006.
[24] DJI, "Agras t16," 2019. [Online]. Available: https://www.dji.com/it/t16
[25] ——, "Flame wheel arf kit F550," 2019. [Online]. Available: https://www.dji.com/it/flame-wheel-arf/spec
[26] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, Oct. 1960.
[27] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.
[28] A. Fotouhi, M. Ding, and M. Hassan, "Understanding autonomous drone maneuverability for internet of things applications," in *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, 2017, pp. 1–6.
[29] M. B. Milam, R. Franz, and R. Murray, "A new computational approach to real-time trajectory generation for constrained mechanical systems," in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2000.
[30] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.
[31] I. Davidson and S. S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in *Knowledge Discovery in Databases (KDD)*, 2005.
[32] F. R. Bach and M. I. Jordan, "Learning spectral clustering," in *Advances in neural information processing systems*, 2004, pp. 305–312.
[33] K. Goss, R. Musmeci, and S. Silvestri, "Realistic models for characterizing the performance of unmanned aerial vehicles," in *26th International Conference on Computer Communication and Networks, ICCCN*. Vancouver, BC, Canada, July 31 - Aug. 3, 2017, pp. 1–9.
[34] J. Lee, *A First Course in Combinatorial Optimization*. Cambridge University Press, 2004.
[35] E. Lawler, *Combinatorial optimization - networks and matroids*. New York: Holt, Rinehart and Winston, 1976.
[36] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a submodular set function subject to a matroid constraint," in *IPCO*, June 2007.
[37] M. Fisher, G. Nemhauser, and L. Wolsey, "An analysis of approximations for maximizing submodular set functions – II," *Math. Prog. Study*, vol. 8, pp. 73–87, 1978.
[38] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie-Mellon University, Management Sciences Research Group, Tech. Rep., 1976.
[39] "Gurobi," Last accessed on January 2019. [Online]. Available: http://gurobi.com
[40] "Lenovo x3550-m5," Last accessed on January 2019. [Online]. Available: https://www.lenovo.com/it/it/data-center/servers/racks/System-x3550-M5/p/77XS7HV7V38
[41] DJI, "Naza-m v2 compact designed multirotor autopilot system," 2019. [Online]. Available: https://www.dji.com/it/naza-m-v2
[42] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 6753–6760.

Novella Bartolini (SM '16) graduated with honors in 1997 and received her PhD in computer engineering in 2001 from the University of Rome, Italy. She is now associate professor at Sapienza University of Rome. She was visiting professor at Penn State University for three years from 2014 to 2017. Previously, she was visiting scholar at the University of Texas at Dallas for one year in 2000. She was program chair and program committee member of several international conferences. She has served on the editorial board of Elsevier Computer Networks and ACM/Springer Wireless Networks. Her research interests lie in the area of wireless networks and network management.

Andrea Coletta received his MSc degree in Computer Science form Sapienza University of Rome in Italy, where he is now attending his PhD in Computer Science. His research interests include wireless networks and mobile network, with particular focus on communication and coordination protocols for aerial drones.

Gaia Maselli is an assistant professor at the Department of Computer Science at Sapienza University of Rome, Italy. She holds a Ph.D. in computer science from the University of Pisa, Italy. Dr. Maselli's current research interests concern design and implementation aspects of mobile networks and wireless communications systems, with particular focus on drones, back-scattering networks and Internet of Things. Other interests include design and performance evaluation of networking protocols for RFID systems. Dr. Maselli serves as member of the TPC of several international conferences, is associate editor of Elsevier Computer Communications journal, and serves as reviewer for several journals, such as IEEE TMC, IEEE TPDS, IEEE TWC, IEEE TON. Dr. Maselli participated to many European Community research projects. She is member of IEEE.

Ala' Khalifeh received the PhD degree in Electrical and Computer Engineering from the University of California, Irvine -USA in 2010. He is currently an Associate Professor in the Communication Engineering department at the German Jordanian University. Dr. Khalifeh is the recipient of the Fulbright scholarship (2005-2007). He has received numerous academic and technical awards such as: The Young AFCEA 40 Under 40 (2017).The German Jordanian University award for the industrial relationship (2016) and the German Jordanian University Excellence in Research Award (2015). His research is in communications technology and networking with particular emphasis on optimal resource allocations for multimedia transmission over wired and wireless networks and Wireless Sensor Networks.