Swarm Path Planning for the Deployment of Drones in Emergency Response Missions

A. Anastasiou[1], P. Kolios[1] C. Panayiotou[1] and K. Papadaki[2]

***Abstract*— The rapid development of Unmanned Aerial Vehicles (UAV) technologies over the recent years has been decisive for their integration in emergency response missions. While initial use by first responders focused on manual operations, the need to improve utilization necessitates higher levels of automation.**

Contributing towards that end-goal, this work derives swarm path planning algorithms that can effectively and efficiently be employed to search and monitor the operating field. A swarm is composed by two or more units, that coordinate to achieve the mission objectives including minimizing search time while ensuring coverage of the field. A graph theoretic approach is followed to model the underlying swarm path planning problem and mathematical programming is employed to describe a number of important variants of the cooperative strategies that arise. Thereafter, 4 algorithms are derived to solve the swarm path planning problem that are computationally efficient to implement and use in practice. A thorough performance evaluation is conducted to understand the advantages and disadvantages of each heuristic using a number of key performance metrics.

***Index Terms*— Path planning, drone swarms, emergency response, mathematical programming**

I. Introduction

Among the most important tasks, in any emergency response mission, is to establish situational awareness and search for survivors on the ground as soon as possible. To date, both tasks have been carried out by first responders that scan the affected area on foot carrying out visual inspections in the process. Search operations have always been the most significant and difficult task in a response mission mainly due to the constraint accessibility to the field and harsh environmental conditions. As such, searching the field is by far the most resource hogging processes of a response; involving a significant part of the human resources available on the ground.

Unmanned Aerial Vehicles (UAVs) have the potential to achieve this task both faster and safer, while minimizing the extend of human resources that need to be devoted [1]. Emergency management experts have already realized the many potential gains that these UAV platforms can offer, and begun integrating UAVs operations in their response workflows [2].

Undoubtedly, UAVs have transform the way emergency operations are carried out. With today's technology, UAVs can easily access hard to reach areas, support a variety of on-board sensors to provide a real-time picture of the situation on the ground [3], track the evolving and dynamic situation [4], and enable the distribution of supplies to those in need [5]. UAVs are already used to provide situational awareness to first responders. During the recent wildfires in California, Los Angeles Fire Department used UAVs to assess the damages caused by the Creek Fire near Sylmar and the Skirball Fire in the Bel Air and to recognize hotspots that could potentially reignite. While UAV operations, to date, have only been limited to manual flights, the need for a more systematic and extended operation has driven demand for automated functionalities, including searching across large areas using a fleet of UAVs and monitoring the field for changes.

Our work, is concerned with path (re)planning algorithms employed by a centrally-controlled fleet of UAVs in an emergency response mission. Since time is the most critical performance indicator, it is important to search the whole area as soon as possible. Flight routes are computed in a way as to minimise the overall response time while maximizing the coverage of the field taking into account the battery resource limitations of individual units. Clearly, computed routes can dynamically be updated over time to accommodate for modelling uncertainties and environmental changes. The main contributions of this work can be summarised as follows:

- Mathematical programming formulations are derived for the multi-agent searching problem and for a number of important variations to this problem.
- The hardness inherent with the derived formulations led to the development of alternative novel heuristic algorithms that are shown to perform well under real-time constraints.
- Performance analysis is done through extensive simulations over a board of relevant metrics to demonstrate the applicability of the proposed solution.
- Empirical evaluation of the proposed algorithms and their performance is seen through a series of real-life test flights.

The rest of the paper is organized as follows. Section II summarises the related work. Section III derives mathematical problem formulations for coordinated search paths for emergency response. Section IV describes heuristics that solve the baseline problem efficiently and Section V includes simulation results.

[1]A. Anastasiou, P. Kolios and C. Panayiotou are with the KIOS Research and Innovation Centre of Excellence (KIOS CoE) and the Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, 1678, Cyprus. [2]K. Papadaki is with the Department of Mathematics, London School of Economics and Political Sciences. E-mail:{anastasiou.antreas, pkolios, christosp}@ucy.ac.cy, k.p.papadaki@lse.ac.uk

II. Related Work

To date extensive literature has looked into the problem of path planning for area coverage by individual UAV agents. To model the problem, variants of the traveling salesman and vehicle routing problems have been considered [6] [7], [8] while heuristic [9] [10], and meta-heuristic algorithms [11] [12] [13] [14] have been proposed to solve practical instances of the problem with short computation times. To further improve on the execution times, various decomposition techniques have also been investigated [15] [16]. To deal with uncertainty in the model, a number of studies have looked into stochastic modelling formulations with partially observable Markov decision processes being the most prominent example, [17].

Research on coordinated fleets of UAVs has focused predominantly on addressing the technological shortcoming of individual agents including limited battery, and short communication and sensing ranges [18] [19] [20] [21] [22] [23].

In this paper, we investigate how coordinated mobility control actions can be made to generate search paths for each of a fleet of UAVs to follow so that searching can be split between the agents and carried out in a parallel fashion so as to improve response times. It should be emphasized here that this problem is significantly different than what previous studies have focused on as is evident form the reported literature. Moreover we derive novel mathematical programming formulations of the problem and computationally efficient heuristic algorithms that are shown to perform well in real-time under various realistic scenarios.

III. Problem Formulation

As previously emphasized, the aim is to compute routes for each of $k = 1, \ldots, K$ agents (UAVs) to rapidly search the field. To address this problem, a graph theoretic approach is followed to: 1) model the affected area as a complete directed weighted graph G where arc costs represent flight times, and 2) use G to construct routes that will cover the field without exceeding flight time constrains.

An illustrative example of a complete grid graph G is presented in Fig. 1. Starting from a location s, considered as the source, each agent visits nearby nodes that are within the remaining flight time and returns to s. The challenge is to build such routes in a way that the search area is maximised while the total traversal cost in terms of flight time is minimised.

Let the set of agents be denoted by $\mathcal{K} = \{1, 2, \ldots, K\}$. Graph $G = (N, A)$ is assumed to be a complete graph where $N = \{1, 2, \ldots, n\}$ is the set of nodes of G that we would like to visit (i.e., search) by one or more agent and A is the set of arcs of G. Hence, $(i, j) \in A$ means that a UAV can move from node i to node j and this has flying time $c_{ij} \geq 0$. We assume that if $(i, j) \in A$ then we also have $(j, i) \in A$ with $c_{ij} = c_{ji}$.

Also let d_{ij} indicate the Euclidean distance from node i to node j and v the speed by which UAVs traverse the graph. In this paper we assume that weather conditions allow v to be constant. The corresponding arc cost $c_{ij} = \frac{d_{ij}}{v}$ is then assumed to be the total time needed to traverse the particular arc of the graph. We let $B(k)$ and the total flight time that UAV k can achieve based its initial battery level.

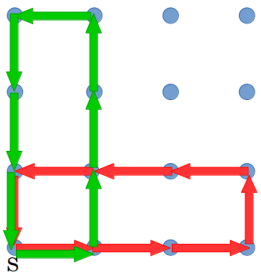


Fig. 1: Feasible tours performed by two UAVs starting at source node s and returning to s within their remaining flight time.

Each agent is initially located at a source node $s \in N$ and its initial remaining flying time is $B(k)$. We would like to find a route for each k that starts at the source node s visits some other nodes in N and returns to s with flying time not exceeding $B(k)$, ensuring the safe return of the UAV back to the source. We assume that the route of each k can repeat nodes and edges and the routes of different UAVs can also overlap in nodes and edges. Thus, the feasible routes of each UAV is a closed walk that goes through s but to simplify notation we call them *tours*.

In this section we define two variations of the problem to address particular mission objectives. These objectives either minimise flight times (battery lives) or maximise the number of nodes visited. For each one of the aforementioned problems we provide mathematical programming formulations in the form of integer or mixed integer linear programs.

A. Minimising Flight Time - P1

The first problem we consider, $(P1)$, has the objective of minimizing the total flying time (battery consumption). Further, we would like each node to be visited by some UAV at least once. A mathematical programming flow formulation of $(P1)$ is shown below:

$$(\mathrm{P1})\ \min \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \tag{1}$$

$$\text{s.t.} \sum_{(i,j)\in A} c_{ij} x_{ij}^k \leq B(k) \quad \forall\, k \in \mathcal{K} \tag{2}$$

$$\sum_{k=1}^{K} \sum_{j:(i,j)\in A} x_{ij}^k \geq 1 \quad \forall\, i \in N \tag{3}$$

$$\sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = 0 \quad \forall\, k \in \mathcal{K}, i \in N \tag{4}$$

$$u_i^k - u_j^k + 1 \leq M(1 - x_{ij}^k) \quad \forall\, k \in \mathcal{K}, (i,j) \in A, i, j \neq s \tag{5}$$

$$x_{ij}^k \in \mathbb{N}, u_i^k \in \mathbb{R} \tag{6}$$

The integer variable x_{ij}^k denotes a flow on arc (i,j) for UAV k indicating the number of times that agent traverses arc $(i,j) \in A$. The objective (1) here is to minimise the total battery consumption for all agents. Constraints (2) ensure that all tours of the UAVs are within their remaining flying time. Constraints (3) ensure that all nodes are visited by at least one UAV by restricting the total outflow from each node $i \in N$ to be at least 1. Note that we allow each UAV to visit a node more than once. The conservation of flow constraints at each node i for each UAV are given in constraints (4). In order to ensure that the flow for each UAV will be a connected cycle that leaves s and returns to s we use the Miller Tuck Zemlin (MTZ) subtour elimination constraints (see [24]) as shown in (5), which are explained below.

The MTZ subtour elimination constraints are used in the well-known Travelling Salesman Problem (TSP) in order to find a minimum-cost Hamiltonian tour (a tour that visits all the nodes exactly once). These constraints eliminate disconnected subtours. In our problem they do something similar for the tour of each UAV. To define these constraints continuous variables u_i^k are used, which are known as node potentials. Also, the parameter M used in the constraint is a large positive integer, which we can set it to be a multiple of the number of arcs in G. For example, suppose there is a cycle $1-2-3-1$ which means that all the flow variables $(1,2)$, $(2,3)$, $(3,1)$ have positive flow and using the MTZ constraints on these arcs we get $u_1 < u_1 + 1 \leq u_2 < u_2 + 1 \leq u_3 < u_3 + 1 \leq u_1$. This is impossible to achieve and thus no such cycle will be formed. However, we allow cycles that have the source s in them because the arcs connected to s are excluded from the MTZ constraints. The conservation of flow constraints on each node along with the MTZ constraints on all nodes except s ensure that each UAV will leave s and return to s. Therefore, constraints (5) construct for each agent a tour that leaves and returns to the source node.

B. Minimising Flight Time Alternative Formulation - P2

It is important to emphasize that in the above formulation the MTZ constraints have been employed to eliminate subtours. However, one drawback of these constraints is that they do not allow certain cycles that could be acceptable UAV routes. For example, in Fig. 2 the route shown is a an acceptable UAV tour, however the MTZ constraints would rule it out because of cycle $6-7-8-12-16-15-14-10-6$; even though this cycle is connected to the source via another cycle.

To address this shortcoming, alternative subtour elimination constraints (ASEC) are designed that only eliminate disconnected subtours. Problem $(P2)$ below reformulates the basic formulation expressed in $(P1)$ to include those ASEC constraints.

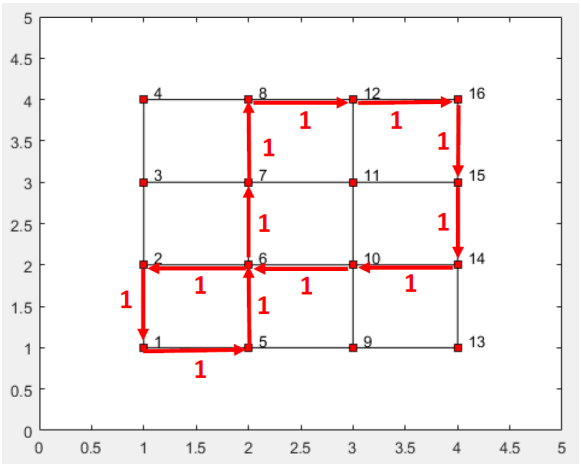


Fig. 2: The depicted tour would have been eliminated by the MTZ constraints because of cycle $6-7-8-12-16-15-14-10-6$ even though it is a valid tour. The number on each arc is the number of times the UAV passes through that arc.

$$\text{(P2)}\ \min \sum_{k=1}^{K} \sum_{(i,j)\in A} c_{ij}^k x_{ij}^k \tag{7}$$

$$\text{s.t.} \sum_{j:(s,j)\in A} x_{sj}^k \geq 1 \quad \forall\, k \in \mathcal{K} \tag{8}$$

$$M \sum_{(i,j)\in A(Q)} x_{ij}^k \geq \sum_{(i,j)\in A'(Q)} x_{ij}^k \; \forall\, Q \subset N, Q \neq \emptyset, k \in \mathcal{K} \tag{9}$$

$$x_{ij}^k \in \mathbb{N}, (2), (3), (4)$$

Constraints (8) ensure that each UAV is connected to s by setting the outflow out of s for each drone to be at least 1. Constraints (9) are the alternative subtour elimination constraints (ASEC). For $Q \subset N$, we define $A(Q)$ to be the set of arcs with only one end in Q and let $A'(Q)$ to be the set of arcs with both ends in Q. The right hand side of the constraint is the total flow within set Q with respect to UAV k and the sum on the left hand side is the total flow in and out of Q. Thus if there is some flow within Q with respect to k then the right hand side is positive which forces the left hand side to be positive which means that the flow in Q is connected to nodes outside of Q. This eliminates disconnected flow cycles but does not eliminate flow cycles that are connected to the source.

C. Maximising Number of Nodes Visited - P3

In the previous formulations, constraint (3) ensured that all nodes are visited by some UAV. This is quite restrictive in the sense that computing tours that visit all the nodes may become infeasible for large instances of graph G since the traversal costs in terms of flying time might be too large for the available UAV resources. In problem $(P3)$ we relax this constraint and instead change the objective to maximise the number of nodes visited. In order to do that we introduce new binary variables z_i that take the value 1 only when some UAV visits node i in its tour, and 0 otherwise. The formulation of

problem $(P3)$ is shown below:

$$(\text{P3}) \max \sum_{i \in N} z_i \tag{10}$$

$$\text{s.t. } z_i \leq \sum_{k=1}^{K} \sum_{j:(i,j)\in A} x_{ij}^k \quad \forall\, i \in N \tag{11}$$

$$\sum_{k=1}^{K} \sum_{j:(i,j)\in A} x_{ij}^k \leq M z_i \quad \forall\, i \in N \tag{12}$$

$$z_i \in \{0,1\} \text{ and } (2),(4),(6),(9)$$

Constraints (11) ensure that if there is no flow of node i with respect to any UAV k then $z_i = 0$ and constraints (12) ensure that if there is some flow out of node i for any UAV then $z_i = 1$.

D. Hardness of Optimal Solutions

Problem $(P1)$ can be easily shown to be NP-hard when we notice that when $k = 1$ it becomes the Travelling Salesman Problem (TSP). Problems $(P2)$ and $(P3)$ on the one hand relax the sub-tour elimination constraints by allowing tours such as the one in Figure 2, but on the other hand they include constraints (9), whose cardinality is exponential in the number of nodes. Thus none of these problems scale well for realistic network instances. In the next section we seek to device heuristic solutions that can be solved efficiently and used in practice to construct UAV tours in real-time.

IV. Practical heuristic algorithms

A. Christofides Variant

Due to several similarities of $(P1)$ and $(P2)$ to the travelling salesman problem (TSP), a variant of Christofides Algorithm (CA), originally published in [25], is initially employed to efficiently compute UAV tours. Christofides algorithm is a constant $\frac{3}{2}$ approximation algorithm. However, our heuristic, called *Christofides Variant* (CV) heuristic, does not have any approximation guarantee.

The idea of the CV heuristic is the following: we first create a spanning tree of G with K branches coming out of s by greedily selecting arcs in a round robin fashion for each UAV k; then each branch subtree is converted to a tour from s by adding to the subtree a minimum cost perfect matching of the odd degree nodes of the subtree, creating an Eulerian circuit and then short-cutting it (as in Christofides algorithm, see [25]). Our problem also has the battery capacity constraints in terms of flying time (see (2)) that we need to satisfy. In order to do that, we keep track of the cost of each subtree. While the subtree cost is within $\frac{B(k)}{2}$ we know that the resulting tour will be within $B(k)$ (see Appendix A); if the subtree cost exceeds this limit then we need to make sure that the resulting tour has cost less than $B(k)$ to ensure that the UAV can return to s.

Let $S_k = \{s\}$ be sets that correspond to each UAV that we will grow at each iteration to form the UAV's tour; let T_k be the current subtree of UAV k, and let $S = \cup_{k=1}^{K} S_k$. Let M_k be a minimum cost perfect matching on T_k. If Q is a set of arcs, then $|Q|$ is the total cost of the arcs in Q.

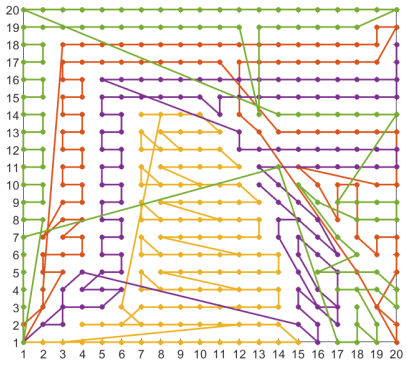
Using a round robin for each UAV in the order $k = 1, \ldots, K$ we add a minimum cost arc (i, j) with $i \in S_k$ and $j \in N \setminus S$ to T_k. Then we update S_k to be $S_k \cup \{j\}$ and also update S. The above steps repeat while $|T_k|$ is less than $\frac{B(k)}{2}$; if adding a new arc (i, j) to T_k results in $|T_k|$ exceeding $\frac{B(k)}{2}$ then the new arc is added only if $|tour(T_k)| \leq B(k)$ (see Appendix A), where $tour(T_k)$ is the tour created as in Christofides algorithm: find a Eulerian tour of the Eulerian graph $T_k \cup M_k$ and shortcut it (remove all nodes except if they appear in the Eulerian tour for the first time) to get $tour(T_k)$.

It is important to note that the subtrees are build concurrently and not one after the completion of another. In the alternative case, creation of the subtrees would have provided a huge advantage to the first UAV in terms of cost but penalize subsequent branches. For instance, lets assume that the first subtree built by the algorithm is for a UAV with a very large battery availability. Then another UAV with less available capacity (i.e., less available flight time) would take on a significantly higher cost tour and thus contribute significantly less in the search process. Hence the computed tours would have significantly higher variance and thus take on much longer to complete. Instead, by extending subtree branches at the same time, tours are more balanced and battery resources more efficiently used. The pseudocode is depicted in Alg. 1.

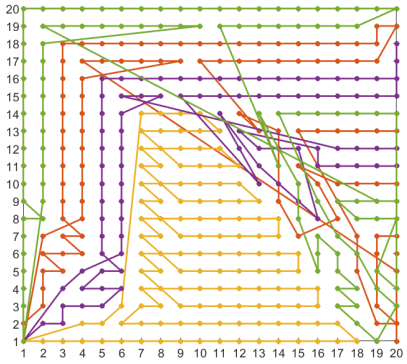
Algorithm 1 Christofides Variant (CV) heuristic

Step 0:
$S_k = \{s\}$; $S = \cup_{k=1}^{K} S_k$; $T_k = \emptyset$, $R = \{1, \ldots, K\}$.
Step 1:
for $k \in R$ **do**
  Find minimum cost arc (i, j) with $i \in S_k$, $j \in N \setminus S$.
  if $|T_k \cup \{(i, j)\}| \leq \frac{B(k)}{2}$ **then**
    $T_k \leftarrow T_k \cup \{(i, j)\}$, $S_k \leftarrow S_k \cup \{j\}$, Update S.
  else
    Compute $tour(T_k)$ as in CA.
  end if
  if $|tour(T_k)| \leq B(k)$ **then**
    $T_k \leftarrow T_k \cup \{(i, j)\}$, $S_k \leftarrow S_k \cup \{j\}$, Update S.
  else
    Remove k from R.
  end if
end for
Step 2:
If R not empty repeat Step 1, else go to Step 3.
Step 3: Compute $tour(T_k)$ for all k as in CA.

We use two variations of the CV heuristic: (1) as above we compute an optimal minimum cost perfect matching and we call that CV-OPT; (2) we use an approximate perfect matching instead and we call that CV-AX. Even though the minimum cost perfect matching problem can be solved in polynomial time by the blossom algorithm by Edmonds (see [26]) and improved versions thereafter, it is still too time consuming for this kind of application. To approximate the



(a) Using CV-OPT heuristic



(b) Using CV-AX heuristic

Fig. 3: Colored tours generated for 4 UAVs using CV heuristics.

optimal matching we use an algorithm from [27] which approximates the optimal matching with a factor $log(n)$ of the optimal solution. We refer the reader to the aforementioned paper for details on the algorithm. The CV-AX will execute in the same fashion as described in Alg. 1 except: (1) when calculating $tour(T_k)$ an approximate perfect matching will be used instead and, (2) only when $|T_k \cup \{(i,j)\}| \leq \frac{B(k)}{1+log(n)}$ we will allow (i,j) to enter T_k, otherwise we recompute the $tour(T_k)$ and check that $|tour(T_k)| \leq B(k)$ (see Appendix A).

To visualise the algorithm's execution, we consider the following setup:

- 400-node grid graph with horizontal and vertical distance $d_{ij} = 100m$
- Fleet size equals to $K = 4$ and $v = 10m/s$
- Mean flight time equal to 25 minutes

Figure 3 plots the tours generated by CV-OPT and CV-AX based on the above setup.

B. Greedy Best (GB) Heuristic

An alternative heuristic approach is proposed in this section. As before, a fleet of K UAVs is considered originating at the source node s. At first, the K nearest nodes to s are added to each corresponding S_k and the corresponding arcs are added to paths P_k. An iterative process follows, where paths are created in parallel by adding a new node to each path based on the lowest cost in a round robin fashion. The

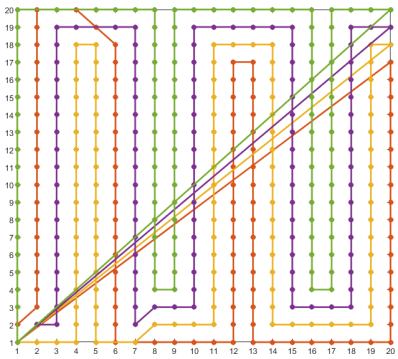


Fig. 4: Colored tours generated for 4 UAVs using GB heuristic.

process continues for each UAV until there is no available flight time remaining except to return to node s or when all the nodes of the graph have been visited. Feasibility is ensured by adding a node to a path only if the remaining flight time is enough to reach s from the next candidate node. We use the notation $l(S_k)$ to denote the last node added to the set S_k. The pseudocode is depicted in Alg. 2.

Algorithm 2 Greedy Best (GB) Heuristic

Step 0:
$S_k = \{s\}$; $S = \cup_{k=1}^{K} S_k$; $P_k = \emptyset$, $R = \{1, \ldots, K\}$.
Step 1:
for $k \in R$ **do**
  Find minimum cost arc (i,j) with $i \in S_k$, $j \in N \setminus S$.
  if $|P_k \cup \{(i,j)\}| + c_{js} \leq B(k)$ **then**
    $P_k \leftarrow P_k \cup \{(i,j)\}$, $S_k \leftarrow S_k \cup \{j\}$, Update S.
  else
    Remove k from R.
  end if
end for
Step 2:
If R not empty repeat Step 1, else go to Step 3.
Step 3:
The tours are $P_k \cup \{(l(S_k), s)\}$ for each k.

Figure 4 demonstrates an instance of the algorithm execution assuming the same setup as in Fig. 3.

As can be seen from the figure, the expansion of each path stops whenever the remaining flight capacity is not enough to visit any additional nodes in the graph. At that point the arc directly connecting the final node of the path to the source node is used for the return of the UAV and thus a feasible tour is created.

C. Dual Path (DP) Heuristic

Contrary to the GB heuristic where one path is created and then directly connected to s to create a tour, the DP heuristic constructs forward and return paths concurrently and then connects them.

Two paths are constructed for each UAV where nodes are added to each path only if the corresponding UAV has enough flight time to cover the arc needed to connect the

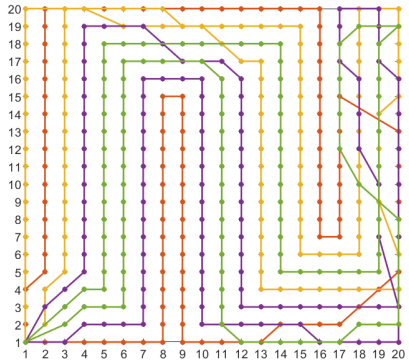


Fig. 5: Colored tours generated for 4 UAVs using DP heuristic

two paths assigned to it. As before the algorithm terminates when all nodes are visited or when there is no more available capacity of the paths to expand. In either case, the two paths of each UAV are joined to form a tour and the heuristic terminates. Algorithm 3 details the steps followed. We denote by P_k^1 and P_k^2 the outbound and return paths that are developed by the heuristic.

Algorithm 3 Dual Paths Algorithm

Step 0:
$S_k^1, S_k^2 = \{s\}$; $P_k^1, P_k^2 = \emptyset$; $R = \{1, \ldots, K\}$;
$S = \cup_{k=1}^{K} S_k^1 \cup S_k^2$.
Step 1:
for $k \in R$ **do**
  Find minimum cost arc (i, j) with $i \in S_k^1$, $j \in N \setminus S$.
  if $|P_k^1| + c_{ij} + |P_k^2| + c\left(j, l(S_k^2)\right) \leq B(k)$ **then**
    $P_k^1 \leftarrow P_k^1 \cup \{(i, j)\}$, $S_k^1 \leftarrow S_k^1 \cup \{j\}$, Update S.
    Find minimum cost arc (p, q) with $p \in S_k^2$, $q \in N \setminus S$.
    if $|P_k^1| + |P_k^2| + c_{pq} + c\left(l(S_k^1), q\right) \leq B(k)$ **then**
      $P_k^2 \leftarrow P_k^2 \cup \{(p, q)\}$, $S_k^2 \leftarrow S_k^2 \cup \{q\}$, Update S.
    else
      Remove k from R.
    end if
  else
    Remove k from R.
  end if
end for
Step 2:
If R not empty repeat Step 1, else go to Step 3.
Step 3:
The tours are $P_k^1 \cup P_k^2 \cup \{\left(l(S_k^1), l(S_k^2)\right)\}$.

An instance of the Dual Path algorithm is demonstrated in Fig. 5 using the same setup as in Fig. 3.

As can be seen, tours created by DP do not resemble either the behavior of the CV or GB tours and a balance is maintained between the forward and reverse legs of the tour.

V. Simulation Setup and Results

As emphasized above, the derived mathematical programming formulations are hard to solve in practical settings under real-time constrains and thus the performance evaluation conducted hereafter focuses only on the solutions provided by the 4 heuristics derived in the previous section. All 4 heuristics aim at computing flight tours with the least travelling cost (flight time) while maximizing the search space covered (number of nodes visited).

A. Simulation Scenario Setup

All simulations were conducted in Matlab on a standalone tower pc hosting an i5 processor and 8GB of RAM. We used a complete graph consisting of 600 nodes that models a field of 6 square kilometers in size. Two shapes of networks were considered: (1) a grid network of 20x30 nodes spread evenly over a rectangular area of 6 square kms; (2) a random network where the location of 600 nodes where randomly generated inside the same area. The source node s in both of the above networks was the bottom left corner point of the rectangular area. The size of the selected area was chosen in such a way to allow various levels of coverage (percentage of nodes visited) for each heuristic. Each UAVs speed was set to $v = 10m/s$ and fleet size was varied to 4, 5 or 6 UAVs. Note that the maximum flight time for numerous consumer fix-wing and multi-rotor UAVs is approximately 45 minutes. In our experiments we randomly generated initial remaining flying time $B(k)$ for each UAV: we used 100 Monte Carlo simulations, where in each simulation we drew $B(k)$ uniformly from the interval $[20, 30]$ minutes thus giving an average flight time of 25 minutes.

B. Coverage Comparison and Computational Times

Figure 6 depicts in boxplots the percentage of nodes visited (coverage percentage) by the 4 heuristics both for the 20x30 grid network and the randomly generated network for 4, 5 and 6 UAVs. The variation in the boxplots shown below is from the randomly generated $B(k)$'s (see section V-A). The GB approach clearly outperforms both CV-OPT and CV-AX and slightly outperforms the Dual Path algorithm. Indicatively, for a fleet of 5 UAVs in the grid network, GB achieves 100% coverage, while CV-OPT achieves only 86%, CV-AX 84%, and DP 97%. The difference is even more apparent in the random network with GB giving full coverage with 5 UAVs whereas the CV-OPT, CV-AX and DP giving 63%, 63%, 95%, respectively. Even though the two CV heuristics use a higher level of sophistication than the greedy fashion of GB, in terms of coverage GB clearly outperforms them. DP is comparable to GB and as we will see later it has some other advantages.

The average computational times are shown in Fig. 7(a); they were performed on a 20x30 Grid Network and the averages are shown (in log scale) over 100 realisations of UAV flight times drawn uniformly from [20,30] minutes. As expected, the greedy heuristics GB and DP have much lower computational times; with GB having the lowest execution times. Hence, GB outperforms all the heuristics with respect to coverage and running time. Low running times are important for this kind of application since it is likely that the tours might need to be recalculated in real time if the underlying conditions change. The difference in coverage between CV-OPT and CV-AX is not very high

and thus using an approximation algorithm to compute the perfect matching in CV-AX reduces the computational time tremendously without significantly affecting coverage. The computational times of GB and DP are comparable with GB performing slightly better.

C. Algorithm Depended Hardware Safety

In this section we evaluate the quality of the flying tours computed by the 4 proposed algorithms. One important aspect to consider in the quality of the tours, is how safe computed tours are when the mission needs to be aborted. For instance, in case of a change in the environmental conditions or aircraft faults or sudden drops in battery levels, the UAVs might need to abort their mission and thus it would be preferable that the UAVs are not too far from the source for too long. To evaluate the tours computed by the different algorithms against such a metric, the following measures are derived: (1) the average distance from each node on the tour to the source node (see figure 7(b)); (2) the distance from s of the first node where the battery level has dropped below $x\%$ of the initial battery (for $x = 30\%$ see fig. 7(c); for $x = 25\%$ see fig. 7(d)). When looking at Fig. 7(b), we can see that all the heuristics average to about the same distance from s. However, when looking at Figs. 7(c) and 7(d) we can see that when the battery is low, the GB tours place the UAVs far from the source; with CV-AX and CV-OPT performing slightly better, whereas DP computed tours having approximately half the distance. This is also apparent in Figs. 3(a), 3(b), 4, 5 where example tours are shown for each of the heuristics: the GB tours get further and further away from s and then comes back to s in one long arc, whereas the DP tours are already on their way back when the battery runs low. In this respect, the DP algorithm strikes a good performance tradeoff between coverage, running time and robustness of the computed tours. Importantly, using the proposed DP algorithms, would ensure that during the critically low battery regions, the UAVs would be closer to the source hence their return to the source is more likely to be successful.

VI. Empirical Evaluation

To demonstrate the applicability of the proposed solution, a hardware implementation has been conducted using the following system architecture. As shown in fig. 8(a) the system consists of two main hardware components, namely the command and control (C2) workstation and the UAV-side remote controller. ROS (Robotic Operating System) acts as the middleware responsible for the exchange of data between the two sides [28]. On top, the C2 workstation uses a web interface for monitoring the fleet and calls the python code that executes the searching algorithm to periodically recompute search paths for the available UAVs. On the drone side, API (Application Programming Interface) calls are made from a mobile terminal connected to the remote controller and the UAV to retrieve data and set waypoints. APIs calls are made to the flight controller onboard the UAV that take on the task to adjust lower-level control decisions. In our implementation, UAVs manufactured by DJI have been used and thus the developer APIs from DJI have been employed (https://developer.dji.com/).

Figure 8(b) illustrates the web interface including, on the left-hand side, data received from 2 UAVs connected on the system, and on the right-hand side, the search paths computed for the two UAVs. For clarity, the underlying map has been removed. In the instance shown, one UAV managed to complete its path and returns to the source while the second UAV is approaching its last waypoint (graph node) before starting its journey back to the source node. In any case, search path resemble those depicted in fig. 4.

Figure 9 depicts the experimental results of executing the paths computed using the GB and DP heuristic algorithms for a scenario of 2 UAVs covering a 100x100m square area consisting of 50 nodes. The two plots in the figures 9(a) and 9(b) are average values of the results obtained from the two UAVs for each algorithm, respectively for the number of waypoints visited and the total battery depleted.

In the figures, there is a constant rate for both the waypoints visited as well as for the battery depletion over time. Interestingly, for this small network, using the DP heuristic the UAVs manage to visit all nodes faster and, more importantly, with less battery consumption.

VII. Conclusions and Future Work

We consider the problem of searching to achieve fast situational awareness in an emergency response mission using a UAV swarm. Four heuristic algorithms were derived and compared with respect to coverage, running times, and robustness considerations. The proposed GB algorithm maximized coverage while the DP algorithm was shown to strike the best performance with respect to the mean distance from the source.

Future work aims first at advancing the system model by incorporating flight dynamics so as to better estimate battery levels, consider heterogeneous agents with different battery depletion characteristics and consider the effect of searching with possibly conflicting objectives such as tracking after a target has been detected. Furthermore, future work will incorporate weather conditions such as wind, gusts and precipitation to increase the effectiveness and applicability of the algorithms in real-life situations.

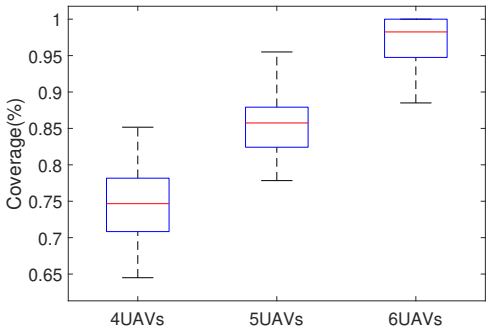
Acknowledgements

European Union Civil Protection under grant agreement No 783299 (SWIFTERS); European Unions Horizon 2020 research and innovation programme under grant agreement No 739551 (KIOS CoE); Republic of Cyprus through the Directorate General for European Programmes, Coordination and Development.

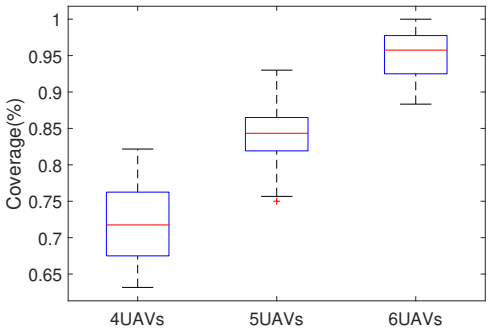
References

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, pp. 460 EP –, 05 2015.

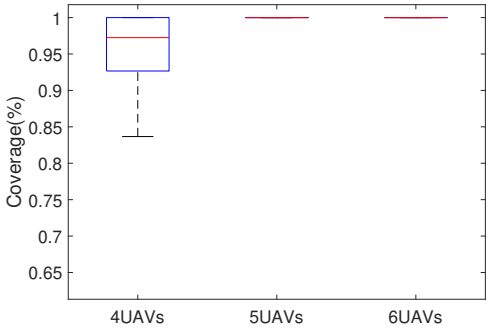
[2] G. De Cubber and et al, *Search and Rescue Robotics - From Theory to Practice*. InTech, Aug. 2017. [Online]. Available: https://doi.org/10.5772/intechopen.68449



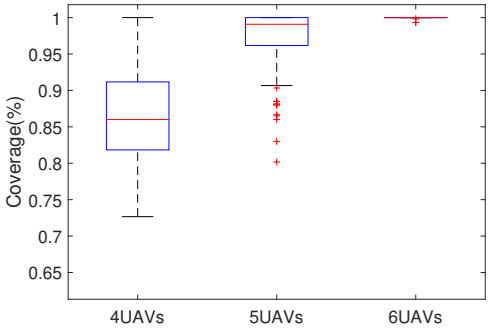
(a) CV-OPT on a Grid Network



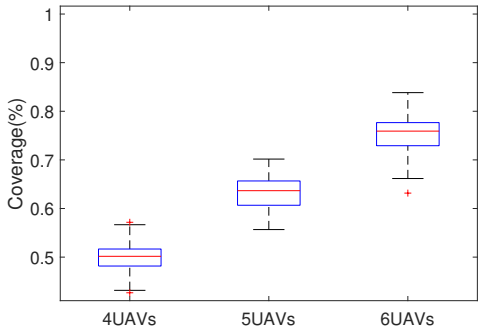
(b) CV-AX on a Grid Network



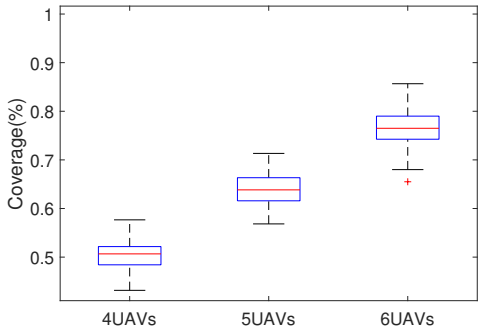
(c) GB on a Grid Network



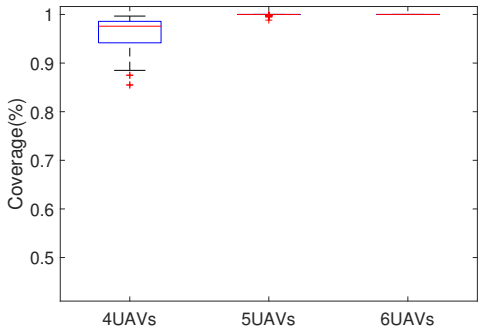
(d) DP on a Grid Network



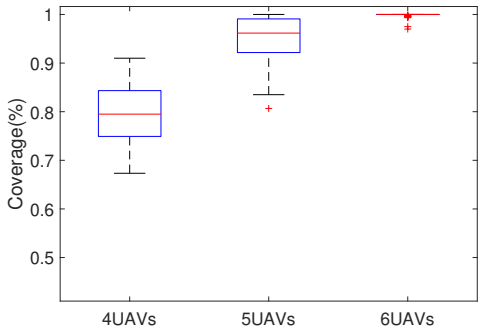
(e) CV-OPT on a Random Network



(f) CV-AX on a Random Network

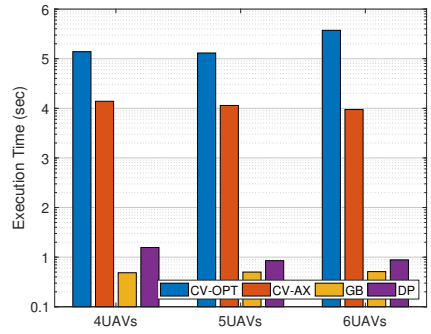


(g) GB on a Random Network

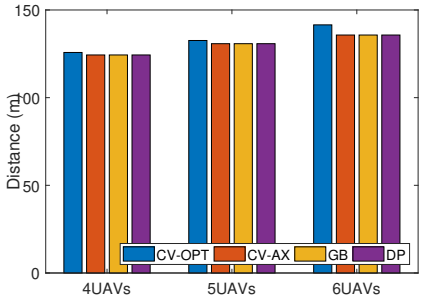


(h) DP on a Random Network

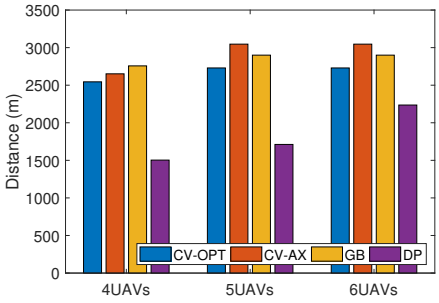
Fig. 6: Average percentage of nodes covered: in a 20x30 grid network for (a)-(d) and in a 600 node randomly generated network for (e)-(h). Data of Box Plots come from 100 realisations of UAV flight times that are drawn from the uniform distribution in the range of 20min-30min of flying time. Each figure shows the coverage for 4, 5 and 6 UAVs.



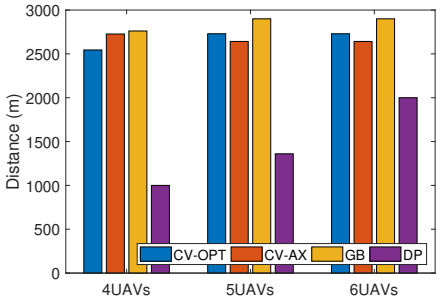
(a) Average execution times in log scale.



(b) Mean distance from each point in the tour to the source node



(c) Mean distance from the first node in the tour that battery level is below 30% of initial battery



(d) Mean distance from the first node in the tour that battery level is below 25% of initial battery

Fig. 7: (a)-(d) were performed on a 20x30 Grid Network with average values shown over 100 Monte Carlo simulations.

[3] K. A. Ghamry and Y. Zhang, "Fault-tolerant cooperative control of multiple uavs for forest fire detection and tracking mission," in *2016 3rd Conference on Control and Fault-Tolerant Systems (SysTol)*, Sept 2016, pp. 133–138.

[4] T. T. C. P. S. Papaioannou, P. Kolios and M. Polycarpou, "Probabilistic search and track with multiple mobile agents," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2019, pp. 1, keywords=, doi=, ISSN=, month=Jun,.

[5] N. Al Theeb and C. Murray, "Vehicle routing and resource distribution in postdisaster humanitarian relief operations," *International Transactions in Operational Research*, vol. 24, no. 6, pp. 1253–1284. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12308

[6] E. Yanmaz, R. Kuschnig, M. Quaritsch, C. Bettstetter, and B. Rinner, "On path planning strategies for networked unmanned aerial vehicles," in *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, April 2011, pp. 212–216.

[7] H. Bai, S. Shao, and H. Wang, "A vtol quadrotor platform for multi-uav path planning," in *Proceedings of 2011 International Conference on Electronic Mechanical Engineering and Information Technology*, vol. 6, Aug 2011, pp. 3079–3081.

[8] H. Wang, B. Yan, X. Li, X. Luo, Q. Yang, and W. Yan, "On optimal path planning for uav based patrolling in complex 3d topographies," in *2016 IEEE International Conference on Information and Automation (ICIA)*, Aug 2016, pp. 986–990.

[9] J. Hu, L. Xie, J. Xu, and Z. Xu, "Multi-agent cooperative target search," pp. 212–216, Jun 2014.

[10] S. Waharte and N. Trigoni, "Supporting search and rescue operations with uavs," in *2010 International Conference on Emerging Security Technologies*, Sept 2010, pp. 142–147.

[11] S. Konatowski and P. Pawowski, "Ant colony optimization algorithm for uav path planning," in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, Feb 2018, pp. 177–182.

[12] D. Zhang, Y. Xian, J. Li, G. Lei, and Y. Chang, "Uav path planning based on chaos ant colony algorithm," in *2015 International Conference on Computer Science and Mechanical Automation (CSMA)*, Oct 2015, pp. 81–85.

[13] L. Geng, Y. F. Zhang, J. J. Wang, J. Y. H. Fuh, and S. H. Teo, "Mission planning of autonomous uavs for urban surveillance with evolutionary algorithms," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, June 2013, pp. 828–833.

[14] Y. Bao, X. Fu, and X. Gao, "Path planning for reconnaissance uav based on particle swarm optimization," in *2010 Second International Conference on Computational Intelligence and Natural Computing*, vol. 2, Sept 2010, pp. 28–32.

[15] A. A. Marro and L. M. G. Gonalves, "A path planning method for multi-uav system," in *2013 Latin American Robotics Symposium and Competition*, Oct 2013, pp. 129–135.

[16] T. H. Pham, Y. Bestaoui, and S. Mammar, "Aerial robot coverage path planning approach with concave obstacles in precision agriculture," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*, Oct 2017, pp. 43–48.

[17] G. Murtaza, S. Kanhere, and S. Jha, "Priority-based coverage path planning for aerial wireless sensor networks," in *2013 IEEE Eighth International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, April 2013, pp. 219–224.

[18] E. Yanmaz, S. Yahyanejad, B. Rinner, H. Hellwagner, and C. Bettstetter, "Drone networks: Communications, coordination, and sensing," *Ad Hoc Networks*, vol. 68, pp. 1 – 15, 2018, advances in Wireless Communication and Networking for Cooperating Autonomous Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870517301671

[19] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, "Multi-uav exploration with limited communication and battery," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2230–2235.

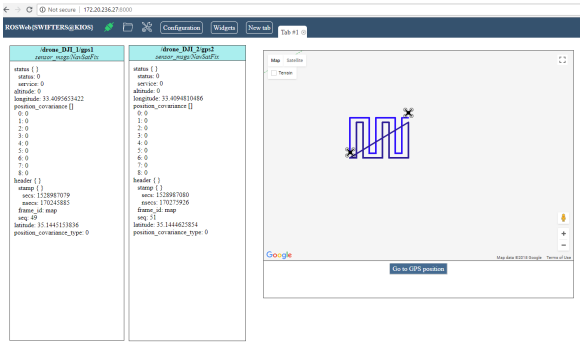
[20] S. Hayat, E. Yanmaz, T. X. Brown, and C. Bettstetter, "Multi-objective uav path planning for search and rescue," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 5569–5574.

[21] J. Modares, F. Ghanei, N. Mastronarde, and K. Dantu, "Ub-anc planner: Energy efficient coverage path planning with multiple drones," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 6182–6189.

[22] J. Scherer and B. Rinner, "Short and full horizon motion planning for persistent multi-uav surveillance with energy and communication constraints," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 230–235.

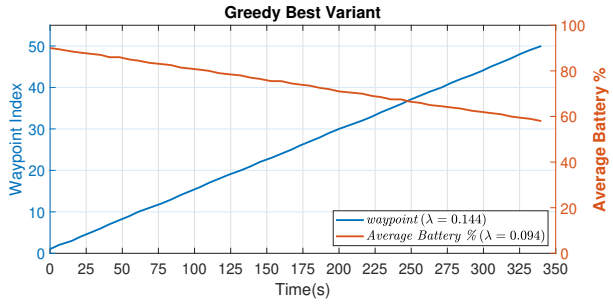
[23] Y. Bouzid, Y. Bestaoui, and H. Siguerdidjane, "Quadrotor-uav opti-

(a) System architecture.

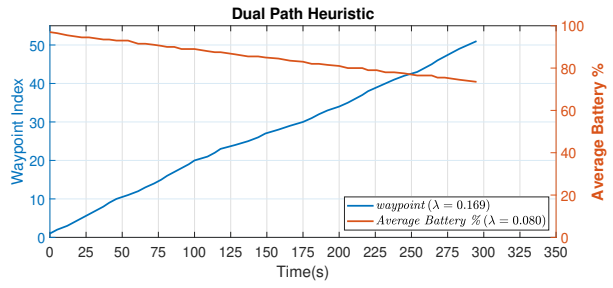


(b) Prototype implementation

Fig. 8: The figures depict the implemented architecture and an instance of the user interface.



(a)



(b)

Fig. 9: Empirical evaluation of the GB and DP heuristics.

mal coverage path planning in cluttered environment with a limited onboard energy," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 979–984.

[24] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulations and traveling salesman problems," *J. ACM*, vol. 7, p. 326329, 1960.

[25] N. Christofides, "Worst-case analysis of a new heuristic for the traveling salesman problem," p. 10, 02 1976.

[26] J. Edmonds, "Paths, trees and flowers," *Canadian Journal of Mathematics*, vol. 17, p. 449467, 1965.

[27] M. Wattenhofer and R. Wattenhofer, "Fast and simple algorithms for weighted perfect matching," *Electronic Notes in Discrete Mathematics*, vol. 17, p. 285291, 2004.

[28] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*, 1st ed. O'Reilly Media, Inc., 2015.

APPENDIX A

Let T be a spanning tree and let M be the cost of the minimum cost perfect matching on the odd degree nodes of T; let $|T|$ and $|M|$ denote the costs of the corresponding objects, and let $|H|$ be the cost of the optimal hamiltonian cycle. We know from the proof that Christofides algorithm is a $\frac{3}{2}$ approximation algorithm that: $|T| \leq |H|$, $|M| \leq \frac{1}{2}|H|$ (see [25]). Since $|H|$ is the optimal cost we also have that: $|H| \leq |T| + |M|$. Combining the above we get:

$$|H| \leq |M|+|T| \leq \frac{1}{2}|H|+|T| \Rightarrow \frac{1}{2}|H| \leq |T| \Rightarrow |M| \leq |T|.$$

Using notation from section IV-A we have $|M_k| \leq |T_k|$. If we also have $|T_k| \leq \frac{B(k)}{2}$, then this means $|M_k| + |T_k| \leq 2|T_k| \leq B(k)$, which ensures that there is enough flying time for the UAV to perform the tour and return to s. Thus, while $|T_k| \leq \frac{B(k)}{2}$ there is no need to check that the current tour satisfies the flying time constraints but when $|T_k|$ exceeds $\frac{B(k)}{2}$ then we need to calculate the cost of the tour and check that it remains feasible when adding the last arc.

In the CV-AX heuristic we use a $log(n)$-approximation algorithm for the perfect matching (see [27]). Suppose this approximate matching is M'_k. Then the resulting tour of UAV k has cost less than or equal to:

$$|M'_k|+|T_k| \leq log(n)|M_k|+|T_k| \leq (1+log(n))|T_k| \leq B(k),$$

where the last inequality holds only if $|T_k| \leq \frac{B(k)}{1+log(n)}$. So we know the UAV is within battery capacity when this last inequality holds.