Solving Infrastructure Monitoring Problems with Multiple Heterogeneous Unmanned Aerial Vehicles

Jakub Ondráček, Ondřej Vaněk and Michal Pěchouček
Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague
Technická 2, Praha 6, Czech Republic
{ondracek, vanek, pechoucek}@agents.fel.cvut.cz

ABSTRACT

A number of critical infrastructures, such as gas or oil pipelines operate in a sensitive environment and any damage done to the infrastructure significantly harms the surrounding fauna and flora and poisons water supplies. The infrastructure thus needs to be monitored and any damage or failure has to be detected and reported as quickly as possible. We focus on the enhancement of the monitoring systems—we propose a first holistic solution for a set of heterogeneous unmanned aerial vehicles (UAVs) which monitor the infrastructure under current technological restrictions such as speed, battery endurance and sensing radius. We solve the problem of (1) the allocation of charging/maintenance stations in the area, (2) the assignment of the UAVs to the stations and (3) the computation of their trajectories with respect to the environment sensitivity. We propose a formal graph-based model capturing problem constraints and requirements. We explore possible decompositions of the problem and we propose a number of algorithms allowing to choose between algorithm runtime and solution quality. The results show that our approach can be used to monitor real-world sized infrastructures of a length of tens of kilometers using up to five UAVs.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems

General Terms

Algorithms; Security

Keywords

Unmanned Aerial Vehicles; Critical Infrastructure Protection; Mixed-Integer Programming; Trajectory Planning

1. INTRODUCTION

In many regions around the world, many critical infrastructures operate using an outdated technology. One of them are for example oil pipeline systems in developing regions, such as in equatorial Africa where failures of parts of the system are on daily order and the leaked oil can be counted in hundreds of thousands of barrels per year and the environmental damage is literally unquantifiable [16]. A number of sensors monitoring the infrastructure is already in place (such as optical sensors or human patrols), however, recent spread of unmanned aerial vehicles (UAVs) allows to deploy UAVs for this task, significantly lowering costs while decreasing damage detection time and allowing efficient damage tracking.

We focus on the problem of the efficient utilization of UAVs for monitoring of elongated infrastructures in environmentally sensitive areas. Current limitation of UAVs lies in their limited flight time—the UAVs need to periodically recharge their batteries to continuously monitor the infrastructure. We thus need to tackle the problem of allocation of charging stations allowing the UAVs to recharge their batteries. We address the problem both on the tactical and on the operational level: we look for an optimal allocation of the stations along the infrastructure and we design a set of monitoring trajectories taking flight constraints of UAVs into account.

Our approach is based on a large body of work which is dedicated to problems of utilization of UAVs for various scenarios, such as search and rescue [7], border patrolling [5], or forest fire monitoring [2]. Domain-independent relevant research directions focus on the energy efficiency of UAVs [22], on trajectory planning [14, 17], or on periodical recharges of already allocated stations [10]. These problems are computationally hard and they are solved in isolation—they do not tackle the station-allocation problem jointly with the trajectory design problem.

We aim to connect all necessary parts together to solve the monitoring problem: (1) we formalize the described problem as a mixed-integer mathematical program capturing the base allocation problem while minimizing weighted age of information under UAV flight constraints, (2) we propose a logical decomposition into partial models, (3) we design a number of algorithms chaining the partial models together, allowing finding the optimal trade-off between the quality of the solution and the runtime of the algorithm. Finally, (4) we evaluate the algorithms on both synthetic and real-world data to demonstrate its capabilities.

The results show that the complete problem is hard to solve and the scalability with respect to the main parameters of the problem, such as the number of agents, the number of stations and the flight time of UAVs, is very limited. Proposed algorithms—exploiting decomposition and iterative approach—are more scalable, able to solve problems of

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May, 4–8, 2015, Istanbul, Turkey.*

realistic size with up to five of UAVs distributed into 3 stations and monitoring the infrastructure of a total length of $30km$ spanning over $8km^2$, which we demonstrate on the pipeline system spanning within and around the Rostock harbor.

2. BACKGROUND

With the increasing stability, availability and decreasing costs of unmanned aerial systems with multiple unmanned aerial vehicles, we can observe their deployment in many surveillance problems [2, 5, 7]. The UAVs are typically equipped with a number of sensors able to detect malfunction of the infrastructure together with specifying the exact place of the failure. The UAVs are also utilized to detect unauthorized intruders in the monitored area, however—as such case is fundamentally different from nature-caused failures—we do not address such scenarios in this paper (we recommend work of Tambe et al. [21] to solve similar scenarios).

Currently, the most pressing issue is the energy-inefficiency of UAVs and their restricted maximum uninterrupted flight time. Even though UAVs with endurance in tens of hours exist [1], we focus on civil deployment and affordable solutions where the UAVs offered have endurance in tens or at most a few hundreds of minutes [22]. Considering these flight time constraints, the problem of large infrastructure monitoring becomes a challenging issue.

To solve this issue, it is vital to deploy a number of charging stations throughout the area to allow UAV recharging [6]. The stations are stationary by definition[1] and they cannot be easily reallocated once the UAVs are deployed for every-day area monitoring. We thus face a complex problem of optimal charging station allocation and computation of flight trajectories for UAVs while taking into account properties of the environment and of the monitored infrastructure lying within.

A large body of work is focused on UAV scheduling considering real-world restrictions. Typically, the problem is modeled using mathematical programming approach—Kim at al. [11] developed a MILP scheduling model for multiple UAVs starting from multiple bases and performing a number of tasks in an adversarial area. In subsequent work [10] the authors introduce multiple shared recharge stations placed within the area, however, they do not address the problem of station distribution. We follow a similar modeling approach; however, we focus on the problem on the holistic level.

Another approach was taken by Shima and Schumacher [19] who propose a linear program for the task scheduling based on the vehicle routing problem. They schedule tasks for a heterogeneous set of UAVs with differing endurances and a single shared base positioned in space without considering properties of the environment with respect to tasks and without planning waypoints for the flight trajectories.

Another highly relevant body of research is focused on trajectory planning for one or more UAVs, typically formalized as a movement on a graph. Nigam and Kroo [14] solve the problem of a persistent surveillance with a focus on creating a trajectory for UAVs with respect to the aircraft dynamics. They formalize the problem as a patrolling algorithm for a single or multiple UAVs and they design an efficient complete algorithm for solid planar graphs and propose a heuristics for other types of planar graphs. Pasqualetti et al. [17] solve the problem of cooperative patrolling. Their work is similar to ours in patrolling a set of points with priorities, however, their approach is not directly applicable because they look only for solutions with non-intersecting trajectories and they do not deal with charging station allocation. We borrow optimization criterion from Chevaleyre [3] who solves a similar patrolling problem on a graph, minimizing time lag between two visits of each node. The described papers always solve only a subset of our problem, and their environment representation and set of constraints is not compatible with our requirements to be directly extended.

We base our work on preliminary results by Ondráček et al. [15] who focused on the trajectory planning problem over a planar graph with multiple UAVs, however, without taking the allocation of charging stations into account and having an unnecessarily complex model. We reuse parts of the notation and we extend the utility function which is now reusable across various domains, we integrate the base allocation problem into the model, we significantly reduce the complexity of the model by considering edge-based variables, we consider multi-step recharging of UAVs and we develop a new set of decomposition algorithms able to solve the model significantly faster.

3. FORMAL MODEL OF THE PROBLEM

We abstract the specific problem of multi-UAV monitoring of pipeline system into a problem of the multi-agent monitoring of an elongated infrastructure in a heterogeneous environment. The model can be then applied to a number of scenarios, such as UAV-based monitoring of water, gas or dangerous liquid pipelines and power lines or multi-USV (unmanned surface vehicles operating on the surface of the water) monitoring under-water pipelines and other structures.

The problem is formalized as an optimization problem with multiple mobile agents optimizing a joint criterion function on a graph representing given infrastructure. In following sections, we explain the environment representation, we explicitly define the movement of agents within the environment and we formulate the utility function to be optimized as the aggregate cost of possible damage in time.

3.1 Environment representation

In our model the monitored infrastructure is represented as a graph $G(N, E)$ where E is a set of edges which represent monitored segments of the given structure and N is a set of nodes connecting the segments.

In the graph G any two neighboring nodes n_i and n_j are connected by two opposite directed edges $e(n_i, n_j)$ and $e(n_j, n_i)$. In the mathematical model, for brevity, we denote the opposite edge of e as $\bar{e}$. Additionally, in each node n, a loop edge $\lambda(n)$ is added to allow waiting in the node (e.g., in a base). The loop edge does not have an opposite edge.

The discretization of the time dimension is tied to the infrastructure discretization: the restriction posed is that any agent can move over any edge in an integer number of time steps. The planning horizon is thus divided into T time steps where all time steps have an equal duration τ (e.g., 10 minutes).

[1]With mobile recharging stations, we face similar problem of recharging the mobile charging stations using larger static stations.

Finally, as an integral part of the problem, we require to find a set of nodes $B \subset N$ of size $|B| = \beta$. These nodes represent the recharge stations for the agents, further referred as bases.

3.2 Agent Model

We formalize the set of UAVs monitoring the infrastructure as a set of agents K. The agents are moving along the edges of the graph. We introduce a binary *edge-entry* variable ${}^k a_e^t$, which is set to 1 when an agent k *enters* the edge e in the time step t; capturing the edge entry time step allow us to take into account various speeds of agents. We use an indirect representation of agents' speeds: we define a parameter ${}^k\sigma_e$ which denotes how many time steps the agent k requires to move over the edge e. All loop edges can be traversed by any agent in a single step, i.e., ${}^k\sigma_{\lambda(n)} = 1$ for all nodes $n \in N$ and all agents $k \in K$.

Each agent $k \in K$ has a set of parameters which reflects its physical restrictions: speed (defined above), endurance ${}^k D$ and recharge ratio ${}^k R$. ${}^k D$ defines the maximum amount of flight time (in minutes) an agent can spend monitoring without recharging. In the mathematical model, we introduce the *available flight time* variable ${}^k d^t$ for all agents and time steps $t \in T$, which represents for how many minutes the agent k can move without recharging.

The recharge rate ${}^k R$ represents how fast the energy of the agent k is replenished (measured again in minutes). The agent can recharge itself only in its own base[2].

The available flight time of an agent is updated in each time-step t in the following way: if the agent is not located in its base in the time-step t, the available flight time is decreased by τ (duration of the time step). If the agent is located in its base, it is partially recharged, thus the available flight time is increased by the recharge rate ${}^k R$. The agent k is located in its base if it is on the loop edge $\lambda(n)$ of a node $n \in B$. The assignment of the agent k to a base node n is encoded in the *base assignment* binary decision variable ${}^k b_n$ set to 1 if the agent k is assigned to the base in the node n. The available flight time of the agent k in the time step t is computed as:

$$ {}^k d^t = \begin{cases} \min\{{}^k d^{t-1} + {}^k R, {}^k D\} & \text{if } \exists n \in N : {}^k a^t_{\lambda(n)} \cdot {}^k b_n = 1 \\ {}^k d^{t-1} - \tau & \text{otherwise} \end{cases} \tag{1} $$

3.3 Cost Function

Given the problem objectives, we aim to minimize the time of all components of the monitored system being unobserved by an agent, weighted by the importance of the component. Formally, we minimize the weighted age of information (AoI) for all edges $e \in E$.

For all time steps $t \in [1, T]$, we can compute a cost of potential damage being caused by an external event or by an internal malfunction from the last visit. If an agent covers an edge in a given time step, AoI of that edge is set to zero. If no agent covers the edge, AoI is increased in each step by τ. Agents are always moving over edges of the given graph, an agent can move over an edge multiple time steps (given by the parameter ${}^k\sigma_e$) and in every time step, each agent covers exactly two edges e and $\bar{e}$ or a single loop edge $\lambda(n)$. By the edge coverage, we understand the ability of the agent to sense any damage on a given edge through its sensor. The coverage of an edge is expressed by the *edge coverage* variable γ_e^t which is defined for all edges $e \in E$ as:

$$ \gamma_e^t = min\left(1, \sum_{k \in K} \sum_{i=0}^{\sigma_e^k - 1} \left({}^k a_e^{t-i} + {}^k a_{\bar{e}}^{t-i} \right) \right) \tag{2} $$

I.e. the edge e is covered if some of the agents entered the edge e or the opposite edge $\bar{e}$ in $t - \sigma_e^k$ time steps before t or later.

The age of information for the edge e at the time step t is defined recursively as:

$$ AoI_e^t = \begin{cases} 0 & \text{if } \gamma_e^t = 1 \\ AoI_e^{t-1} + \tau & \text{otherwise} \end{cases} \tag{3} $$

Using AoI and properties of the monitored system, we can define the cost of the expected damage c_e^t for each node at each time step:

$$ c_e^t = AoI_e^t \cdot s_e \cdot f_e \cdot i_e \tag{4} $$

The cost for an edge is given by the age of information multiplied by the size (or diameter) of the component s_e the edge represents, by its importance i_e[3] and by the failure rate f_e. For a given problem instance, s_e, f_e and i_e are fixed (and typically, we assume the same values for two opposite edges e and $\bar{e}$). We can then replace the multiplication term by a constant parameter C_e, which can be possibly replaced for different scenarios by another parameter supplied by subject matter experts. We also set $C_e = 0$ for all loop edges $\lambda(n), n \in N$. By combining Equations (3) and (4) we get the current cost for the edge e in the time step t:

$$ c_e^t = \begin{cases} 0 & \text{if } \gamma_e^t = 1 \\ c_e^{t-1} + C_e \cdot \tau & \text{otherwise} \end{cases} \tag{5} $$

And we can express the aggregated cost for the system and the time horizon T as:

$$ C = \sum_{t \in T} \sum_{e \in E} c_e^t \tag{6} $$

3.4 Mathematical Program

We integrate all requirements and constraints into a single mathematical program to create a complete model of the problem which can now be specified as a problem of finding positions of bases which can be placed on any node $n \in N$, distribution of all agents into the bases and finding trajectories of the agents respecting constraints on their movement while minimizing the cost defined by Equation 6:

$$ \min \sum_{t \in T} \sum_{e \in E} c_e^t \tag{7} $$

$$ c_e^0 = 0 $$

[2]In some models, the agents can recharge in any base [19]. Due to maintenance problems with current UAVs and potential problems with recharge conflicts, we assign each UAV to a single base. However, multiple agents can share a single base. In principle it is straightforward to change or model to reflect the requirement for shared bases.

[3]Environmental Sensitivity Index (ESI)[8] is typically used in the oil pipeline domain. It defines sensitivity of the environment w.r.t. possible oil damage on a scale from 1 to 10 where 10 is the highest environment sensitivity.

$$\forall e \in E \tag{8}$$

$$c_e^t \geq c_e^{t-1} + C_e \cdot \tau - M \cdot \gamma_e^t \quad \forall e \in E, \forall t \in [1,T] \tag{9}$$

$$\gamma_e^t \leq \sum_{k \in K} \sum_{i=0}^{\sigma_e^k - 1} \left({}^k a_e^{t-i} + {}^k a_{\bar{e}}^{t-i} \right) \quad \forall e \in E, \forall t \in [1,T] \tag{10}$$

$$\sum_{e \in out(n)} {}^k a_e^t \leq {}^k a_{\lambda(n)}^{t-1} + \sum_{\substack{e \in in(n) \\ e \neq \lambda(n)}} {}^k a_e^{\max\{t-\sigma_e^k;0\}} \quad \forall k \in K, \forall n \in N, \forall t \in [1,T] \tag{11}$$

$$\sum_{e \in E} {}^k a_e^t \leq 1 \quad \forall k \in K, \forall t \in T \tag{12}$$

$$\sum_{e \in E} {}^k a_e^0 = 0 \quad \forall k \in K \tag{13}$$

$$\sum_{e \in out(n)} {}^k a_e^1 = {}^k b_n \quad \forall k \in K, \forall n \in N \tag{14}$$

$$\sum_{e \in in(n)} {}^k a_e^{T-\sigma_e^k} = {}^k b_n \quad \forall k \in K, \forall n \in N \tag{15}$$

$${}^k d^t \leq {}^k d^{t-1} - \tau + ({}^k R + \tau) \cdot {}^k a_{\lambda(n)}^t \cdot {}^k b_n \quad \forall t \in [1,T], \forall k \in K, \forall n \in B \tag{16}$$

$$m_n \geq \frac{1}{|K|} \cdot \sum_{k \in K} {}^k b_n \quad \forall n \in B \tag{17}$$

$$\sum_{n \in N} m_n \leq \beta \tag{18}$$

$${}^k a_e^t \in \{0;1\} \quad \forall e \in E, \forall k \in K, \forall t \in T \tag{19}$$

$$c_e^t \in R \quad \forall e \in E, \forall t \in T \tag{20}$$

$$\gamma_e^t \in [0;1] \quad \forall e \in E, \forall t \in T \tag{21}$$

$${}^k d^t \in \{0; {}^k D\} \quad \forall k \in K, \forall t \in T \tag{22}$$

$${}^k b_n \in \{0;1\} \quad \forall k \in K, \forall n \in B \tag{23}$$

$$m_n \in \{0;1\} \quad \forall n \in B \tag{24}$$

Criterion (7) directly minimizes the cost function as defined in Equation (6). Equations (8–10) describe the advancement of cost on edges in time as described in section 3.3.

The next set of equations ensure validity of agents' paths: Equation (11) guarantees continuity of agents' movement—it defines the movement of the agent k from $\lambda(n)$ or any incoming edge $e \in in(n)$ to any outgoing edge of $e \in out(n)$ of n. Equation (12) defines that the agent k is in the time step t located on at most one edge; Equations (13) and (14) initialize the position of the agents: in the time step $t = 0$ agents are not located on any edge. In the time step $t = 1$, agents have to be located on any edge leading from their base (including a loop edge $\lambda(n)$, allowing waiting in a base). Equation (15) defines that the agent has to be on an edge $e \in in(n) | {}^k b_n = 1$ at ${}^k\sigma_e$ time steps before the end of the planning horizon such that it lands in the base (i.e. node $n | {}^k b_n = 1$) in the time step T.

Equation (16) expresses the computation of agents' endurance if the agent k is in the time step t located on the loop edge of its base (${}^k a_{\lambda(n)}^t \cdot {}^k b_n = 1$) then its flight time variable ${}^k d^t$ is increased by ${}^k R$ otherwise is decreased by τ[4].

The restriction on allocation of bases is expressed by Equations (17) and (18). We solve the restriction using an auxiliary binary variable m_n which is set to 1 if at least one agent has its base in node n; more formally $m_n \geq \min\{\sum_{k \in K} {}^k b_n; 1\}$. Equation (17) is the linearization of this expression. If at least one agent has its base in node n then the right side of the equation is greater than zero; the multiplication with $1/K$ guarantees that value of the right side is less or equal to 1 and m_n has to be set to 1. Equation (18) captures a restriction guaranteeing that the number of unique bases is no greater than β.

Finally, we assume the agents to resolve potential conflicts in their flight plan: i.e., when trajectories of two or more agents intersect in space and time, a collision-avoidance algorithm is used to locally resolve the conflict [20]. If this approach is not suitable, an additional constraint:

$$\sum_{k \in K}^{k} a_e^t \leq 1 \quad \forall t \in T, e \in E \tag{25}$$

can be added and the problem can be resolved directly, however, many good solutions (such as two UAVs flying back and forth along the pipeline and meeting in the middle) can be discarded or the underlying graph has to be extended to allow collision avoidance.

3.5 Problem Size Analysis

In the previous section we formulated a complete model of the problem allowing to solve the problem optimally, however, its scalability—due to an implicit hardness of the problem—is limited. We will explore the structure of the solution space of the problem so that we can design decompositions of the problem allowing us to solve the problem faster.

We denote a set of all candidate solutions of the problem as S. A candidate solution is a set of agents' trajectories not violating problem constraints. We can derive the allocation of bases as well as the assignment of agents into bases from the trajectories. By ${}^k S$ we denote a set of all possible trajectories for a single agent k and by ${}^k S_n$ we denote a set of possible trajectories of a single agent k assigned to a fixed base located in n.

[4]To linearize the multiplication of two binary variables ${}^k a_{\lambda(n)}^t \cdot {}^k b_n$, a new auxiliary variable has to be introduced for each node and each time step. This variable is then constrained such that it is always smaller than ${}^k a_{\lambda(n)}^t$ and smaller than ${}^k b_n$.

If we are solving a single-agent problem over a graph with a single fixed base then the set of all candidate solutions is equal to ${}^{k}S_n$.

If we solve a single-agent problem with a single unallocated base, the set of candidate solutions is equal to the sum of candidate solutions with a fixed base in node n over all nodes $n \in N$:

$$S = \sum_{n \in N} {}^{k}S_n \tag{26}$$

If we solve the multi-agent problem with a set of fixed bases B (however, with agents unassigned to bases), the set of candidate solutions is equal to the Cartesian product of candidate solutions for each agent ${}^{k}S$.

$$S = \prod_{k \in K} {}^{k}S = \prod_{k \in K} \sum_{n \in B} {}^{k}S_n \tag{27}$$

For the multi-agent problem with unallocated bases, the candidate solution space is even larger, as we have to consider all $\binom{N}{\beta}$ possible bases allocations:

$$S = \binom{N}{\beta} \prod_{k \in K} \sum_{n \in B} {}^{k}S_n \tag{28}$$

Considering the hardness of the single-agent problem [4, 9] and the analysis of the solution space above, a clever decomposition of the problem is called for. We follow a natural way and we decompose the problem allowing us to find solutions for real-world sized problems.

4. PROBLEM DECOMPOSITIONS

From the analysis in the previous section, we can see that there are three main aspects which significantly increase the size of search space: degrees of freedom caused (1) by the unallocated bases, (2) by the agents not assigned to bases and (3) by all possible combinations of trajectories of all agents. In following sections, we introduce partial models which are able to solve each of these problems separately and provide inputs to the subsequent partial models. We formalize the description of our algorithms as:

$$ALG(INPUT) \rightarrow SOLUTION$$

where ALG is either a name of the mathematical model which is solved using a standard mixed-integer linear program (MILP) solver or a name of a designed algorithm, $INPUT$ is a set inputs for the algorithm and $SOLUTION$ is a solution of the problem, i.e., a set of bases or the assignment of agents into bases or computed trajectories for each agent.

For the complete model, we thus have

$$CM(G, \beta, K, T, D, R, \sigma, {}^{*}B, {}^{*k}B, {}^{*}S) \rightarrow B, {}^{k}B, S$$

i.e., the complete model has as the input a graph G, a number of bases β, a set of agents K, a number of time steps T, endurances of agents D, their recharge ratios R and the number of time steps for an agent to traverse each edge σ as the input[5]. The complete model can be solved with additional optional parameters denoted with an asterisk—a set of potential bases ${}^{*}B, |B| \leq \beta, B \subseteq N$, the assignment of the agents into bases expressed by a vector ${}^{*k}B = (n_i, n_j, \dots), |{}^{k}B| = |K|$ and a set of trajectories ${}^{*}S, |S| < |K|$ for a subset of agents. The optional parameters fix a subset of decision variables and thus speed up the computation of a solution.

The algorithm solving the complete model returns a set of trajectories S (reconstructed from variables ${}^{k}a_e^t$) and—if not specified by the optional parameters—the set of bases B and the distribution of the agents into the bases ${}^{k}B$ (reconstructed from variables ${}^{k}b_n$).

In following sections, we introduce three partial problems: Base Allocation Problem (BAP), determining positions of the bases, and Trajectory Planning Problem (TPP) with a fixed set of bases and agents assigned to bases. Additionally, we deal with the joint computation of all agent's trajectories by exploiting submodularity of the problem [13] and by solving the CM with fixed bases iteratively in a greedy manner (ICM). We use the solution provided by ICM to either assign agents into bases or directly, as a final output of the algorithm.

4.1 Bases allocation Problem (BAP)

The Bases allocation problem solves the problem of allocating bases to specific nodes in the graph while taking into account endurance of the agents, i.e.,

$$BAP(G, \beta, D, \sigma) \rightarrow B.$$

BAP is equivalent to the weighted maximum set coverage problem where the goal is to maximize the sum of coverage of fixed, time-independent costs C_e of edges.

Before the problem is solved, for each node n a set of reachable edges M_n is precomputed—M_n contains all reachable edges given agents' maximal endurance and speed

$$D^* = \max\{{}^{k}D, \forall k \in K\} \qquad \sigma^* = \max\{{}^{k}\sigma \forall k \in K\}$$

The edge $e(n_i, n_j)$ is reachable from the node n if the shortest path from the node n to both n_i and n_j is shorter than one half of endurance of agent with ${}^{*}D$. We use the depth-first search algorithm to find all edges in M_n for all $n \in N$. For each edge e and each node n, we introduce a binary membership parameter m_e^n set to 1 if $e \in M_n$ and to zero otherwise.

The base allocation problem is formulated as follows:

$$\max \sum_{e \in E} C_e \cdot y_e \tag{29}$$

$$\min(1; \sum_{n \in N} m_e^n \cdot b_n) \geq y_e \qquad \forall e \in E \tag{30}$$

$$\sum_{n \in N} b_n \leq \beta \tag{31}$$

$$y_e \in R \qquad \forall e \in E \tag{32}$$

$$b_n \in \{0, 1\} \qquad \forall n \in N \tag{33}$$

Equation (29) is the criterion maximizing the weight of coverage edges' static costs C_e (computation of C_e is described in section 3.3) using binary variable y_e representing if the edge e is reachable or not, given the allocation of bases represented by a binary decision variable b_n. The relationship between y_e and b_n is given in Equation (30). Equation (31) restricts the number of bases to be at most β. The output of the model is the allocation of bases B captured by the decision variables b_n.

[5]We omit the indexes to make the notation more compact. See section 3 for the exact description.

4.2 Iterative CM (ICM)

The iterative version of the CM (ICM) is equivalent to the CM model with fixed bases except we solve the complete model iteratively with respect to the agents: the algorithm is based on the fact that our problem is submodular in the set of agents. The submodularity of the problem can be seen from the following informal reasoning: the cost function is composed only from non-negative coverage components, guaranteeing that adding a new agent into the set of agents cannot increase the cost of solution. Additionally, trajectories of a larger set of agents K^L cover the graph better than trajectories of a smaller set of agents $K^S \subset K^L$: adding an agent into K^S decreases the cost function more than adding the agent in K^L. (a straightforward formal proof omitted for space constraints) We thus solve the problem by an iterative greedy algorithm which guarantees $(1 - \frac{1}{e})$-approximation [12]. ICM computes trajectories for all agents by computing a trajectory for a single agent while fixing trajectories for other agents (see Algorithm 1).

$$ICM(G, \beta = |B|, K, T, D, R, \sigma, B) \rightarrow {}^{k}B, S$$

Algorithm 1 Iterative CM

```
1: S ← ∅                                   ▷ The set of trajectories
2: for all k ∈ K do
3:     S ← S ∪ CM(G, β, k, T, D, R, σ, B, S)
4: end for
5: return S
```

4.3 Trajectory Planing Problem (TPP)

The Trajectory Planning Problem (TPP) is similar to the complete model except we substitute variables ${}^{k}b_n$ with the pre-computed allocations of bases (computed, e.g., by BAP) and with the assignments of agents into bases (computed, e.g., by ICM) denoted by the vector ${}^{k}B$.

$$TPP(G, K, T, D, R, \sigma, B, {}^{k}B) \rightarrow S$$

TPP uses the same set of equations as CM:

$$\sum_{e \in out({}^{k}B)} {}^{k}a_e^1 = 1 \qquad \forall k \in K \quad (34)$$

$$\sum_{e \in in({}^{k}B)} {}^{k}a_e^{T-\sigma_e^k} = 1 \qquad \forall k \in K \quad (35)$$

$${}^{k}d^t \leq {}^{k}d^{t-1} - \tau + ({}^{k}R + \tau) \cdot {}^{k}a^t_{\lambda({}^{k}B)} \qquad \forall t \in [1, T], \forall k \in K \quad (36)$$

except the following changes: Equations (14), (15) and (16) are replaced with Equations (34), (35) and (36) respectively. Because the variable ${}^{k}b_n$ is removed, the Equation (22) is removed also.

The sums in Equations (34) and (35) are no longer over all edges from all nodes, however, only over all edges from a single node ${}^{k}B$ and the variable ${}^{k}b_n$ is replaced with a constant. The same holds for the Equation (36) where we already know the assignment of the agent k into a base through the vector ${}^{k}B$. The problem is again solved using a standard MILP solver and we can reconstruct trajectories from the decision variables ${}^{k}a_e^t$. BAP, TPP and ICM are used in the following decompositions to speed up the solution process.

4.4 BT Decomposition

In the first Base-Trajectory (BT) decomposition, the problem is decomposed in a straightforward manner into two parts: base allocation problem and computation of trajectories for agents together with their distribution into bases. The algorithm has the following form:

$$BAP(G, \beta, D, \sigma) \xrightarrow{B} CM(G, \beta = |B|, K, T, D, R, \sigma, B) \rightarrow S$$

BAP computes the allocation of bases to specific nodes and the solution is injected into the CM which then solves the assignment of agents into bases and computes their trajectories at once. This decomposition significantly reduces the state space of the problem and the scalability is significantly improved, however, we lose optimality guarantees.

We further enhance the scalability of the algorithm by using ICM to find agents' trajectories:

$$BAP(G, \beta, D, \sigma) \xrightarrow{B} ICM(G, \beta = |B|, K, T, D, R, \sigma, B) \rightarrow S$$

We refer to this decomposition as BTi.

4.5 BDT Decomposition

The fine-grained Base-Distribution-Trajectory (BDT) decomposition starts with the base allocation problem, however, prior the trajectory computation; we solve the distribution of agents into allocated bases. As in the BT decomposition, we use BAP to compute allocation of bases. Subsequently we use the ICM to solve the complete model with the injected base allocation, however, only the distribution of agents into bases ${}^{k}B$ is used as the input to the TPP which computes agents' trajectories:

$$BAP(G, \beta, D, \sigma) \xrightarrow{B} ICM(G, \beta = |B|, K, D, R, \sigma, B) \xrightarrow{{}^{k}B} TPP(G, K, T, D, R, \sigma, {}^{k}B) \rightarrow S$$

5. EVALUATION

In the evaluation process, we focus on two main performance aspects: runtime of algorithms and quality of solutions. The scalability of the model with respect to parameters is measured first, pinpointing the main bottlenecks of the model. We compare the runtime of the four main algorithms: CM and three decompositions (BT, BTi, BDT). For the computation of solution quality we use the value of the objective function of the model. Thus, the lower the value, the better is quality of the solution.

Finally, we demonstrate the algorithms on a real word scenario covering oil pipeline systems of the Rostock Harbor. To solve the MILP models, we use CPLEX 12.5.1. and we execute all algorithms and the solver on 16-core PC with 64 GB RAM.

5.1 Synthetic Tests

For the evaluation, unless stated otherwise, we have used the following default configuration: the cost values are sampled randomly from a thresholded normal distribution $D = min(10, max(1, \mathcal{N}(5, 2)))$, σ is sampled uniformly from a set of values $\{1, 2\}$. The default scenarios have randomly generated planar graphs with 100 nodes. We considered two

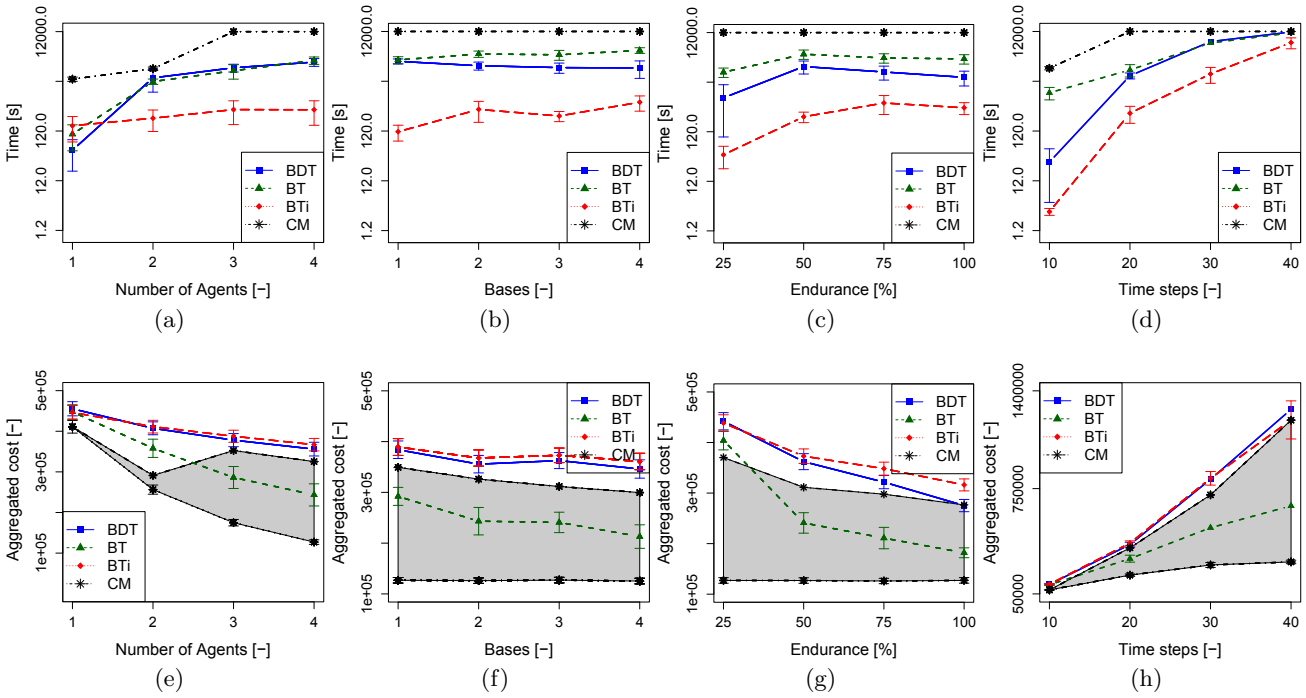

Figure 1: Dependency of the algorithms' runtime (upper row) and quality (lower row) on the main parameters: (a,e) number of agents, (b,f) number of bases, (c,g) endurance of agents and (d,h) number of time steps. The quality of the algorithms is measured in the aggregated cost—the lower the value, the better the solution.

bases in the graph, planning horizon of $T = 20$ time steps and endurance ${}^{k}D$ equals to 10 time steps for all agents. Every instance of scenario was repeated 20 times with random parameters re-sampled. For all problem instances, the maximal runtime was capped on 12000 seconds - the time limit was reached only by the Complete Model. In such case, we display the quality of the relaxed solution together with the value of the currently reached solution (which serve as lower and upper bounds of the optimal solution respectively) as a gray band in the charts quantifying quality.

Figure 1 shows two rows of graphs quantifying the dependency of the runtime of different algorithms and quality of the computed solutions on the main parameters of the problem—number of agents, number of bases, endurance of agents (measured as the percentage of the time horizon) and number of time steps. Figure 1a demonstrates the overall trend in algorithm runtime: the fastest algorithm is BTi, the slowest is CM which is true with respect to all parameters, BDT and BT have a similar runtimes, showing a very small overhead of ICM used to compute the assignment of agents into bases in the case of BDT and demonstrates a non-negligible pruning capabilities of the CPLEX solver.

The number of agents (Figure 1a) significantly impacts the runtime of all algorithms up to a point, where new agents do not bring any major improvement over a solution with a lower number of agents (i.e., we have more agents than it is actually needed to cover the graph). BTi scales linearly with the number of agents as it uses the iterative approach for computation of trajectories.

The algorithms scale well with the number of bases as shown in Figure 1b—the base allocation problem does not dominate the runtime of the algorithms. The endurance of agents has a similar trend as the number of agents (Figure 1c)—up to some point, the runtime increases however, once the agents do not need to recharge (endurance is higher than 50%), the runtime stabilizes when increasing the endurance higher.

The runtime of algorithms is independent on the total size the graph as the state space size of the problem is limited by the endurance of agents. However the runtime of all algorithms is highly dependent on the length of the planning horizon (Figure 1d); the runtime is exponential w.r.t the number of time steps.

The quality of solutions differs for different algorithms. The optimal solution is reached only by the Complete Model (CM). However, CM is unable to find the optimal solution in the limited computation time on most problem instances. We demonstrate the suboptimality of each decomposition using Figure 1e. For two agents and default settings, the quality of the solution computed by CM is on average 256283, BT is 38% worse than CM, BDT is 59% worse than CM and by BTi is 60% worse than CM with the overall lowest quality. The difference between CM and BT shows the suboptimality of the decomposition of the model into base allocation and trajectory computation. The difference between BT and BDT shows the suboptimality of computing the assignment of agents separately from specific trajectories. Finally, the difference between BT and BTi shows the suboptimality of the greedy approach to the trajectory computation. For our problem, the small quality difference between BDT and BTi shows the high importance of agent assignment to bases over specific trajectory planning.

When measuring the quality of the solution when increasing the number of bases and the endurance of agents, we

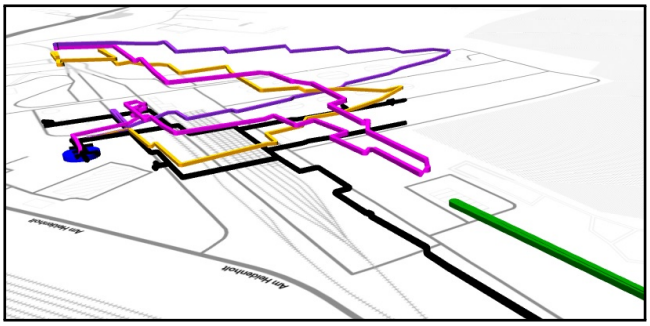

Figure 2: Solution for the Rostock Harbor pipeline system—a detail depicting trajectories of three UAVs assigned to a single base (best viewed in color).

were not able to get the optimal solution (Figures 1f and 1g respectively). Even though we are able to see a significantly better performance of BT. In Figure 1f we can see that with the number of bases (and with the fixed number of agents), the cost decrease is lower than with the increased endurance, leading to the conclusion that the endurance of the UAVs plays a significant role in the area monitoring (it is desirable to reach all parts of the infrastructure).

We can see from the synthetic evaluation that the most scalable approach is BTi, however, its quality is approximately 74 % worse than CM on comparable instances. The size of the time horizon, i.e., the number of time steps, is the main bottleneck (for all algorithms), however, it can be mitigated using a rolling horizon approach [18].

5.2 Monitoring Pipelines in Rostock Harbor

We tested our approach on a real-world dataset of Rostock's Harbor oil pipeline system gathered from the OpenStreetMap[6]. The configuration of the scenario is following: we have $|K| = 5$ UAVs with ${}^kD = 30$ minutes endurance that are located in two bases and the selected area is patrolled for one hour ($T = 60$). The pipelines are approximately $30km$ long and cover an area of $8km^2$. The graph of the infrastructure has 483 nodes and 1427 edges. We solve this task with the fastest algorithm, BTi: the solution was found in 764 minutes. Values on edges are sampled randomly from a threshold normal distribution $D = min(10, max(1, \mathcal{N}(5, 2)))$, speeds of UAVs σ^k is sampled uniformly from a set of values $\{1, 2\}$.

Partial, zoomed-in solution can be seen in Figure 2. The infrastructure is depicted in black; the base is a blue circle. Trajectories of agents as colored corridors with a slightly increased flight height every time step to better observe the solution. AGI Cesium[7] and Stamen map design[8] was used to display the solution.

The total aggregated cost without any UAVs is $2.3 \cdot 10^7$. Using the 5 UAVs with our optimized patrolling strategy, the aggregated cost was decreased by approximately by $7 \cdot 10^6$ to $1.6 \cdot 10^7$ and the UAVs were able to cover 1295 out from 1427 edges (90.7%) during the patrol.

[6] http://www.openstreetmap.org/
[7] https://cesium.agi.com/
[8] http://maps.stamen.com/toner-lite/

6. SUMMARY

The UAV technology has a great potential in monitoring various systems such as critical infrastructures spanning over large areas and sensitive environments. We propose the first holistic approach to utilize a group of UAVs for monitoring graph-like systems. We solve the problem of charging station allocation, assignment of UAVs to the stations and computation of agents' monitoring trajectories. Optimal solutions are hard to find even on small problem instances, however, proposed decompositions and an iterative greedy approach allow to find acceptable solutions for real-world infrastructures.

7. ACKNOWLEDGMENTS

This research was funded by the Office of Naval Research Global (grant no. N62909-11-1-7034). Facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme Projects of Large Infrastructure for Research, Development, and Innovations (LM2010005), is greatly appreciated.

8. REFERENCES

[1] C. Bolkcom. Homeland security: Unmanned aerial vehicles and border surveillance. DTIC Document, 2004.

[2] D. W. Casbeer, D. B. Kingston, R. W. Beard, and T. W. McLain. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science*, 37(6):351–360, 2006.

[3] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004.(IAT 2004)*, pages 302–308, 2004.

[4] Y. Elmaliach, N. Agmon, and G. A. Kaminka. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4):293–320, 2009.

[5] A. R. Girard, A. S. Howell, and J. K. Hedrick. Border patrol and surveillance missions using multiple unmanned air vehicles. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 1, pages 620–625, 2004.

[6] R. Godzdanker, M. J. Rutherford, and K. P. Valavanis. Islands: a self-leveling landing platform for autonomous miniature uavs. In *Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on*, pages 170–175, 2011.

[7] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey. Supporting wilderness search and rescue using a camera-equipped mini uav. *Journal of Field Robotics*, 25(1-2):89–110, 2008.

[8] E. R. Gundlach and M. O. Hayes. Vulnerability of coastal environments to oil spill impacts. *Marine technology society Journal*, 12(4):18–27, 1978.

[9] H.-M. Ho and J. Ouaknine. The cr-uav problem is pspace-complete. *arXiv preprint arXiv:1411.2874*, 2014.

[10] J. Kim, B. D. Song, and J. R. Morrison. On the scheduling of systems of uavs and fuel service stations

for long-term mission fulfillment. *Journal of Intelligent & Robotic Systems*, 70(1-4):347–359, 2013.

[11] Y. Kim, D.-W. Gu, and I. Postlethwaite. Real-time optimal mission scheduling and flight path selection. *Automatic Control, IEEE Transactions on*, 52(6):1119–1123, 2007.

[12] A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 134(6):516–526, 2008.

[13] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions I. *Mathematical Programming*, 14(1):265–294, 1978.

[14] N. Nigam and I. Kroo. Persistent surveillance using multiple unmanned air vehicles. In *IEEE Aerospace Conference*, pages 1–14. IEEE, 2008.

[15] J. Ondráček, O. Vaněk, and M. Pěchouček. Monitoring oil pipeline infrastructures with multiple unmanned aerial vehicles. In *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, pages 219–230. Springer, 2014.

[16] C. O. Orubu, A. Odusola, and W. Ehwarieme. The nigerian oil industry: environmental diseconomies, management strategies and the need for community involvement. *Journal of Human Ecology*, 16(3):203–214, 2004.

[17] F. Pasqualetti, J. W. Durham, and F. Bullo. Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms. *Robotics, IEEE Transactions on*, 28(5):1181–1188, 2012.

[18] S. Sethi and G. Sorger. A theory of rolling horizon decision making. *Annals of Operations Research*, 29(1):387–415, 1991.

[19] T. Shima and C. Schumacher. *Assignment of cooperating UAVs to simultaneous tasks using genetic algorithms*. Defense Technical Information Center, 2005.

[20] D. Sislak, P. Volf, A. Komenda, J. Samek, and M. Pechoucek. Agent-based multi-layer collision avoidance to unmanned aerial vehicles. In *Integration of Knowledge Intensive Multi-Agent Systems, 2007. KIMAS 2007. International Conference on*, pages 365–370. IEEE, 2007.

[21] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

[22] B. Uragun. Energy efficiency for unmanned aerial vehicles. In *Machine Learning and Applications and Workshops (ICMLA), 2011 10th International Conference on*, volume 2, pages 316–320. IEEE, 2011.