Journal of Artificial Intelligence Research 65 (2019) 519-568 Submitted 11/2018; published 08/2019

Autonomous Target Search with Multiple Coordinated UAVs

Chiara Piacentini CHIARAP@MIE.UTORONTO.CA
Department of Mechanical and Industrial Engineering
University of Toronto, Toronto, Canada, ON M5S

Sara Bernardini SARA.BERNARDINI@RHUL.AC.UK
Department of Computer Science
Royal Holloway University of London
Egham, Surrey, UK, TW20 0EX

J. Christopher Beck JCB@MIE.UTORONTO.CA
Department of Mechanical and Industrial Engineering
University of Toronto, Toronto, Canada, ON M5S

Abstract

Search and tracking is the problem of locating a moving target and following it to its destination. In this work, we consider a scenario in which the target moves across a large geographical area by following a road network and the search is performed by a team of unmanned aerial vehicles (UAVs). We formulate search and tracking as a combinatorial optimization problem and prove that the objective function is submodular. We exploit this property to devise a greedy algorithm. Although this algorithm does not offer strong theoretical guarantees because of the presence of temporal constraints that limit the feasibility of the solutions, it presents remarkably good performance, especially when several UAVs are available for the mission. As the greedy algorithm suffers when resources are scarce, we investigate two alternative optimization techniques: Constraint Programming (CP) and AI planning. Both approaches struggle to cope with large problems, and so we strengthen them by leveraging the greedy algorithm. We use the greedy solution to warm start the CP model and to devise a domain-dependent heuristic for planning. Our extensive experimental evaluation studies the scalability of the different techniques and identifies the conditions under which one approach becomes preferable to the others.

1. Introduction

The problem of searching for lost targets has a long history in mathematics and operation research, with the first theoretical studies dating back to the World War II (Koopman, 1946). Since then, the field has continued to evolve and, with the advent of unmanned aerial vehicles (UAVs), autonomous search has become pivotal to numerous real-world applications: from surveillance to border interdiction and law enforcement (Girard et al., 2004), from assisted agriculture (Ammad-Udin et al., 2016) to disaster response (Adams & Friedland, 2011) and protection of wildlife (Hodgson et al., 2016; Bondi et al., 2018), from civil engineering inspections (Liu et al., 2014) to environmental monitoring (White et al., 2008).

When search involves moving targets, the problem is often cast as search and tracking (S&T) because the observer is typically required to track the target to its destination after finding it; we consider S&T here, but focus on the search phase. The tracking phase is handled via a simple reactive controller described in previous work (Bernardini, Fox, Long, & Bookless, 2013).

©2019 AI Access Foundation. All rights reserved.

While the literature on S&T is vast (see Section 7 for an overview), the prevalent method is to formulate search as a path-planning problem and solve it within a *probabilistic* framework (Bourgault, Furukawa, & Durrant-Whyte, 2006; Lavis & Furukawa, 2008; Tisdale, Ryan, Kim, Tornqvist, & Hedrick, 2008; He, Bachrach, & Roy, 2010; Lin & Goodrich, 2014). The probability distribution (PD) of the target location is recursively updated and predicted over time and the search control problem is solved greedily over a very short planning horizon. This strategy has proven successful for short search missions (e.g., a few minutes) over small areas (e.g., one to five square kilometres) with static or predictable targets (Bourgault et al., 2006; Lavis & Furukawa, 2008).

In this paper, we approach the problem of search from a different angle: we focus on the decision-making process to build long-term strategies for the observers to search for the target and on the corresponding action generation to implement such strategies. This aspect has received considerably less attention so far, but it is crucial to improving the search mission performance in realistic domains where the probabilistic approach is too fine-grained to be efficient. We cast S&T as a *deterministic* combinatorial optimization problem consisting of choosing a set of maneuvers for the observers that maximizes the probability of discovering the target. We then use *optimization* techniques to find high-quality solutions efficiently. Existing work in applying generic optimization techniques to S&T is scarce and limited to static targets (Abi-Zeid, Nilo, & Lamontagne, 2011; Morin, Abi-Zeid, Quimper, & Nilo, 2017) or single UAV domains (Bernardini, Fox, Long, & Piacentini, 2016; Bernardini, Fox, & Long, 2017). We use these techniques for hard realistic problems where multiple coordinated observers search for a mobile, evasive target over a large geographical area (around 100 square kilometres) and for an extended period of time (up to 90 minutes).

Our investigation of search emerges from the observation (proven in Section 3) that, in our formulation of the problem, the objective function is *submodular*. Greedy techniques are known to produce bounded approximations when used to maximize submodular functions subject to cardinality (Nemhauser & Wolsey, 1978; Alaei & Malekian, 2010) or other specific types of constraints (Calinescu, Chekuri, Pál, & Vondrák, 2007; Kulik, Shachnai, & Tamir, 2009). In our problem, however, the objective function is subject to a set of temporal constraints that limit the feasibility of solutions, and therefore our greedy algorithm does not enjoy the typical theoretical guarantees. Nonetheless, we implemented the greedy approach and, via an extensive experimental evaluation, we show that, as happens in other applications (Krause & Guestrin, 2011), this technique performs remarkably well in practice, both in general and even more so as the number of available observers increases.

We compare the greedy approach with two different optimization techniques, *AI planning* and *constraint programming* (CP), and explore when it is better to use these more sophisticated strategies. AI planning and CP are mature technologies for which fast solvers are available (e.g. Coles, Coles, Fox, & Long, 2010; Laborie, Rogerie, Shaw, & Vilím, 2018). The advantage of using these generic techniques is that the specificity of the problems resides in the models, while the solvers are generic. Systems for S&T based on these methods are flexible and easily extensible: when the mission changes, the models are modified accordingly, but there is no need to develop different algorithms. Both AI planning and CP, if appropriately configured, are exact approaches: if the solvers are given enough time, they will eventually find the optimal solution. In our scenario, however, time is critical and the solvers are configured to produce a solution within a given time limit, even if the solution is not optimal. Our experiments show that, even when the time available is short, both AI planning and CP outperform the greedy solution, especially when a limited number of search assets are at hand. Both AI planning and CP benefit from the finding of a greedy solution first and using it to guide their search processes. So it is in the integration of the greedy algorithm with more advanced optimization approaches that we find the best solutions for the S&T problem.

Our experimental evaluation is extensive. We study the scalability of different versions of the three optimization techniques as we vary the complexity of the problem, in particular, the number of the possible maneuvers and the number of the observers that can execute them. We identify the conditions under which it is preferable to use one approach instead of the others, in so doing, provide a tool to solve real-world S&T problems in the most effective and cost-efficient way.

1.1 Contributions of the Paper

With respect to previous work on S&T, the novel contributions of this paper are the following:

- As in previous work (Bernardini et al., 2016), we formulate S&T as a deterministic combinatorial optimization problem. We expand the existing formulation by considering search with multiple coordinated observers (Section 2) and we prove several properties of the objective function, most notably that it is submodular (Section 3).
- Leveraging the submodularity of the objective function, we propose a greedy algorithm that is very efficient and exhibits strong performance across all the different settings and problems that we consider (Section 4.1).
- We expand previous work on solving S&T via CP (Bernardini, Fox, Long, & Piacentini, 2017b) in three ways (Section 4.2): (i) we consider missions with multiple UAVs instead of a single observer; (ii) we improve the model by using optional interval variables; and (iii) we use the solution produced by the greedy algorithm to warm-start the CP model. The last two contributions allow us to find substantially better solutions more efficiently than the previous CP approach.
- We also expand an earlier formulation of S&T as a planning problem (Bernardini et al., 2016) to multiple UAVs (Section 4.3) and we exploit the solution provided by the greedy algorithm to formulate a domain-dependent heuristic that significantly improves the performance of the planner we use, i.e. POPF-TIF (Bernardini, Fox, Long, & Piacentini, 2017a).
- Finally, we present a broad experimental evaluation in simulation where these techniques are compared to each other and against existing solutions appropriately expanded to multiple UAVs (Section 5).

2. The Search and Tracking Problem

We consider the following scenario, originally designed in collaboration with our industrial partner, BAE Systems (Bernardini et al., 2013). The target moves on a Euclidean 2-

dimensional (2D) space characterized by a road network and has a distant destination that it wants to reach. The observers are UAVs that can freely move on a 2D plane as we ignore altitude. The UAVs are tasked with following the target to its destination and they are equipped with sensors (cameras in our simulation) to detect the position of the target. Due to the imperfections of the sensors or obstacles in the environment, UAVs can lose sight of the target. The mission starts with one of the UAVs *tracking* the target. When the UAV loses it, for a short period of time (three minutes), the UAV follows the target's predicted trajectory. If the UAV does not recapture the target, it enters a *search* phase to re-locate the target in collaboration with the other UAVs in the team. In every S&T mission, the two phases of searching and tracking alternate until the end of the mission (see Figure 1). We focus on search in this paper.

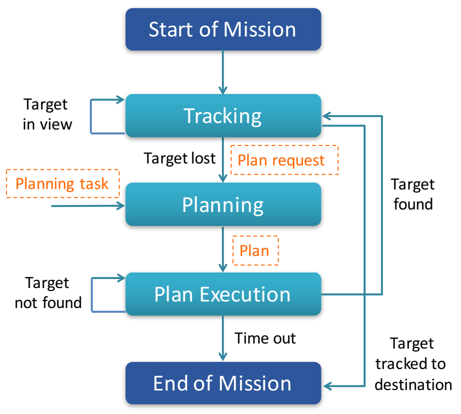


Figure 1: Structure of a typical S&T mission.

As in related work (Paterson, Timmons, & Williams, 2014; Bernardini et al., 2016, 2017b), the UAVs exploit standard flight patterns to search for the target, in particular spirals and lawnmowers (see Figure 2). The spiral pattern is effective for covering areas of high density road network, especially in urban or suburban terrain, while the lawnmower is useful when attempting to search over an elongated stretch covering a major road and including some possible side roads. The advantage of using search patterns is twofold. From a practical point of view, search patterns are predictable maneuvers, easily recognizable by pilots and other flying machines, which facilitates the use of autonomous vehicles by promoting trust in autonomy. From a technical point of view, patterns allow us to model target search as a *combinatorial optimization problem* instead of a continuous optimization one, as it has been done traditionally (Stone, 1975). This approach opens the door to applying combinatorial algorithms to search. The combinatorial problem is to find a set of executable patterns that maximizes the probability of (re)discovering the target while satisfying temporal and resource constraints. We assume that a control unit on the ground runs those algorithms to identify the best patterns and then communicates them to the team of UAVs for execution.[1]

In the rest of this section, we provide a formal definition of the search problem and its objective function, while in the next section we study properties of this objective function.

1. It would be interesting to explore distributed solutions to our problem. We leave that for future work.

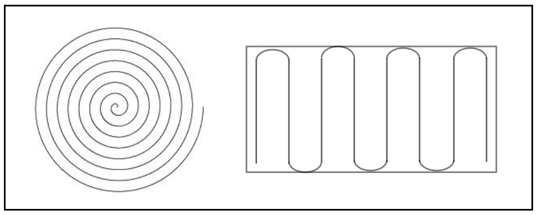


Figure 2: The UAVs exploit standard search patterns to search for the target: spirals (left) and lawnmowers (right).

2.1 Problem Definition

We model the target search as a combinatorial optimization problem: find a set of patterns in specified locations among a pool of candidates that maximizes a closed-form expression that approximates the total probability of discovering the target.[2]

Let $\mathcal{O}$ be a set of observers, all with the same flight and sensing capabilities. We assume that the observers know (i) the road network, (ii) a set of potential destinations $\mathcal{D}$ for the target; and (iii) a set of possible paths Γ that the target might follow to reach the destinations. Each path $\gamma \in \Gamma$ is associated with a probability $PD(\gamma)$ that represents the initial belief of the observers on the target trajectory and depends on the behavioral model of the target. For example, if no information on the target motion is available, a uniform distribution can be used for the paths; if the target has an evasive behavior, the probability distribution can incorporate information on the different concealment levels of the paths (Bernardini et al., 2017b).

Let $\mathcal{C}$ be a set of search patterns laid over specific locations on the map that the observers can execute (*candidates*) and D^o be a matrix for every observer $o \in \mathcal{O}$, with $m+1$ positions, where m is the number of search pattern candidates. Given a matrix D^o, position $(0, \sigma_i)$ is the flight time from the initial position of the observer o to the start point of the pattern σ_i and position (σ_i, σ_j) is the flight time from the end point of the pattern σ_i to the start point of the pattern σ_j. We generate the initial set of patterns $\mathcal{C}$ by running a Monte Carlo simulation (MCS) over the area of operation. In particular, as in previous work (Bernardini et al., 2017), we consider as our search area a circular sector centred on the target's last known position (LKP). The road network is represented as a graph $\mathcal{G} = \langle \mathrm{V}, \mathrm{E} \rangle$, built by discretizing the search area in cells with fixed side-length. The vertexes V correspond to the cells, while the edges are given by pairs of adjacent cells that are connected by at least one road. We select a set of paths in the graph from the LKP to the candidate destinations $\mathcal{D}$ (the shortest path and all those paths whose length differs from the shortest path's length by a customizable factor that represents the concealment level that the target is assumed to adopt) and we simulate the target motion with a standard MCS. The MCS identifies points in the search area that present the highest probability of finding the target at different

2. In previous work (Bernardini et al., 2016), the objective function is a linear combination of the total probability and the expected time to rediscover the target. Here, we consider only the total probability because a preliminary experimental analysis that we conducted in simulation shows that adding the expected time does not change the success rate of finding the target.

points in time. We create candidate patterns that have those points as their centres and can be executed in appropriate time windows when the target can plausibly be in the areas covered by those patterns based on its motion model. For more details on the MCS, the interested reader can consult Appendix A, Bernardini et al. (2017) or Bernardini et al. (2017b).

Each pattern σ is associated with a duration d_σ, which is the time that the UAV takes to execute that pattern, and a time window, i.e., minimum t^-_σ and maximum t^+_σ times at which the observers can start the execution of the pattern σ. Once a pattern is scheduled for execution, we call $\boldsymbol{t} : \mathcal{C} \rightarrow \mathbb{R}$ the function that returns the exact time $\boldsymbol{t}(\sigma) \in [t^-_\sigma, t^+_\sigma]$ at which the execution of σ starts. The value of t^-_σ and t^+_σ are calculated considering the distance between the LKP of the target and σ and the minimum and maximum velocity of the target. Each search pattern $\sigma \in \mathcal{C}$ is also associated with a subset $\Gamma_\sigma \subseteq \Gamma$ called the set of *compatible* paths, i.e., the set of paths that the UAV can observe while executing the pattern σ because they are in its cone of visibility. Finally, we associate a detection probability ϕ_σ to each pattern σ. This is the probability of finding the target in an execution of the pattern σ conditioned on the target having initially chosen any of the paths in Γ_σ. The function ϕ_σ encodes both the randomness in the motion of the target and the sensor noise and is conditioned on the execution of its corresponding pattern σ within the associated time window.

Note that each candidate in $\mathcal{C}$ can be executed more than once. It can be beneficial to execute a pattern multiple times when it covers an area that carries a high probability of discovering the target. We can transform the problem to an equivalent version where each search pattern can be executed at most once, by creating k_σ copies of each pattern σ. The constant k_σ indicates the maximum number of times that σ can be executed. For each pattern σ, k_σ can be easily calculated considering its duration, its window of activation and the number of available observers: $k_\sigma = \lfloor \frac{t^+_\sigma - t^-_\sigma}{d_\sigma + \min_{o \in \mathcal{O}, \rho \in \mathcal{C}} D^o(\rho,\sigma)} |\mathcal{O}| \rfloor$. We call $\widetilde{\mathcal{C}}$ the set of patterns that we obtain by following this procedure and $\widetilde{D}^o$ the updated travel-time matrices.

Given a subset $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ of $\widetilde{\mathcal{C}}$ and a partition of Σ, $\mathcal{P}(\Sigma)$, with at most $|\mathcal{O}|$ elements, we call *observer assignment* a function $S : \mathcal{O} \rightarrow \mathcal{P}(\Sigma)$ that assigns an element of $\mathcal{P}(\Sigma)$ to each observer in $\mathcal{O}$. Given an observer $o \in \mathcal{O}$, we call its (possibly empty) associated partition $S(o)$ a *plan* for o.

Given a subset $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ of $\widetilde{\mathcal{C}}$, we say that Σ is *executable* if we can find a function $\boldsymbol{t}$ and an observer assignment S such that, for each observer $o \in \mathcal{O}$, if we order the elements of $S(o)$ according to $\boldsymbol{t}$, i.e., $S(o) = (s_1, \ldots, s_k)$, where s_i indicates the i^{th} search pattern in the sequence $S(o)$, the following two conditions are satisfied:

- for every two consecutive patterns s_i and s_{i+1} in $S(o)$, with $i = 1, \ldots, |S(o)| - 1$, $\boldsymbol{t}(s_{i+1}) \geq \boldsymbol{t}(s_i) + d_{s_i} + \widetilde{D}^o(s_i, s_{i+1})$;
- $\boldsymbol{t}(s_1) \geq \widetilde{D}^o(0, s_1)$.

The temporal constraints associated to the executability of a sequence of search patterns executed by a UAV (e.g., Figure 3) can be encoded in a Simple Temporal Network (STN) (Dechter, Meiri, & Pearl, 1991), whose consistency can be determined in polynomial time.

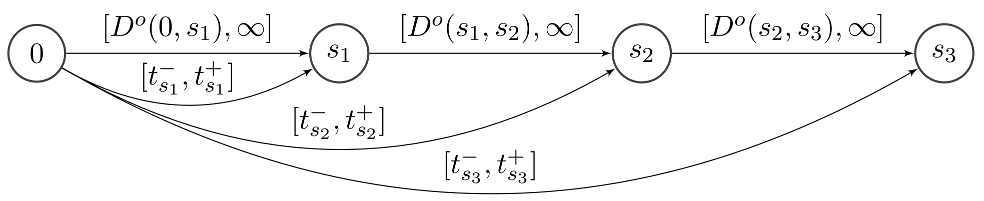


Figure 3: Simple temporal network representing the constraints of a sequence of search patterns $S = (s_1, s_2, s_3)$ executed by an observer $o \in \mathcal{O}$.

Note that, according to our definitions, different observers can execute search patterns simultaneously and the same pattern can be executed by multiple observers at different times or at the same time. Also note that the number of observers to use can vary from 1 to $|\mathcal{O}|$.[3]

We call $P(\Sigma)$ the total probability of finding the target by executing the set Σ. In our model, the total probability depends only by the execution of search patterns and not explicitly on the execution time. Instead, the time-dependency of the target motion is captured by the pre-computed time windows associated with each search pattern candidate, which affect the execution order of the search patterns.

Target search can be formally defined as the problem of finding an executable set of search patterns $\Sigma \in 2^{\widetilde{\mathcal{C}}}$ that maximizes the total probability $P(\Sigma)$.

2.2 The Objective Function

As in Bernardini et al. (2016, 2017b), given a set of patterns $\Sigma \in 2^{\widetilde{\mathcal{C}}}$ and a pattern $\sigma \in \widetilde{\mathcal{C}}$ that is not already in Σ, we can express the objective function $P(\Sigma)$ using the following closed-form expression (see Appendix B for the derivation of all the formulas in this section):

$$\begin{aligned} &P(\Sigma \cup \{\sigma\}) = P(\Sigma) + P_{\Sigma \cup \{\sigma\}*} \cdot (1 - P(\Sigma)) \\ &P(\emptyset) = 0 \end{aligned} \tag{1}$$

This equation gives us a recursive structure to compute $P(\Sigma)$ and tells us that the probability of finding the target by executing $\Sigma \cup \{\sigma\}$ is equal to the sum of the probability of finding the target at the previous step, i.e., by executing Σ, and the probability of finding the target by executing the newly added pattern σ provided that the target has not been discovered earlier, $P_{\Sigma \cup \{\sigma\}*}$. Given a path γ, let $P_\Sigma(\gamma)$ be the probability that the target is following the path γ after the execution of the search patterns in Σ and conditioned on their failure. The probability $P_{\Sigma \cup \{\sigma\}*}$ can then be expressed as follows.

$$P_{\Sigma \cup \{\sigma\}*} = \phi_\sigma \sum_{\gamma \in \Gamma_\sigma} P_\Sigma(\gamma) \tag{2}$$

3. It might be interesting to find an optimal executable set $\hat{\Sigma}$ in $\widetilde{\mathcal{C}}$, defined as an executable set with a partition $\hat{\mathcal{P}}(\Sigma)$ that minimizes the number of observers necessary to execute the patterns in $\hat{S}$ as well as that maximizes the total probability $P(\hat{\Sigma})$. We leave this as future work.

We consider all the paths γ that are compatible with σ and sum the probabilities that the target is following the path γ after the execution of the patterns in Σ and conditioned on their failure, weighting each summand by the probability of detection associated with σ, ϕ_σ. The probability $P_{\Sigma\cup\{\sigma\}}(\gamma)$ that the target is following the path γ conditioned on the failure of the search patterns in $\Sigma \cup \{\sigma\}$ can be expressed by using another recursive equation:

$$P_{\Sigma\cup\{\sigma\}}(\gamma) = \frac{P_\Sigma(\gamma)\cdot(1-\phi_\sigma\cdot \mathbb{1}_{\Gamma_\sigma}(\gamma))}{1-P_{\Sigma\cup\{\sigma\}*}} \tag{3}$$

$$P_\emptyset(\gamma) = PD(\gamma) \tag{4}$$

where $\mathbb{1}$ is the indicator function: $\mathbb{1}_A(x) = 1$ if $x \in A$ and 0 otherwise.

Equation (4) sets the initial probability of the target following a path γ according to our initial hypothesis, $PD(\gamma)$. Once we start executing the patterns, we gain information about the target: any unsuccessful pattern gives us negative information about which path the target is following. As expressed in Equation (3), if the observer has failed to recapture the target when executing a pattern σ from which the path γ is visible, the observers will decrease their confidence that the target is following γ by a quantity that depends on the quality of their sensors, represented by the detection probability ϕ_σ.

Note that (1) can be rewritten as:

$$(1-P(\Sigma\cup\{\sigma\})) = (1-P(\Sigma))\cdot(1-P_{\Sigma\cup\{\sigma\}*}) \tag{5}$$

The recursive definitions of $P(\Sigma)$ and $P(\Sigma)(\gamma)$ can be written as follows.

Proposition 2.1. Given a path γ, let $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a set of search patterns in $\widetilde{\mathcal{C}}$. Then:

$$P_\Sigma(\gamma) = \frac{P_\emptyset(\gamma)\cdot\prod_{i=1}^{n}(1-\phi_{\sigma_i}\cdot\mathbb{1}_{\Gamma_{\sigma_i}}(\gamma))}{\prod_{i=1}^{n}(1-P_{\{\sigma_1,\cdots,\sigma_i\}*})} \tag{6}$$

$$1-P(\Sigma) = \prod_{i=1}^{n}(1-P_{\{\sigma_1,\cdots,\sigma_i\}*}) \tag{7}$$

2.3 Example

We now consider an illustrative example of the problem.

Example 2.1. Figure 4 shows a problem instance with one observer, three destinations d_1, d_2 and d_3, and a path for each destination γ_1, γ_2, γ_3 with initial probability $PD(\gamma_i) = \frac{1}{3}$ and five search pattern candidates $\sigma_i, \forall i = 1, \ldots, 5$. The total detection probabilities and activation time windows of each search pattern are shown in the figure. We assume that the duration of each search pattern is 99 and that the travel time between every position is 1, hence each search pattern can be executed only once.

If we want to maximize the total probability of finding the target, the optimal sequence is $S^* = (\sigma_5, \sigma_4, \sigma_3)$, producing a total probability of $P(S^*) = \frac{2}{3}$. This value is calculated using the recursive Equation (1): when executing the first search pattern σ_5, the probability that

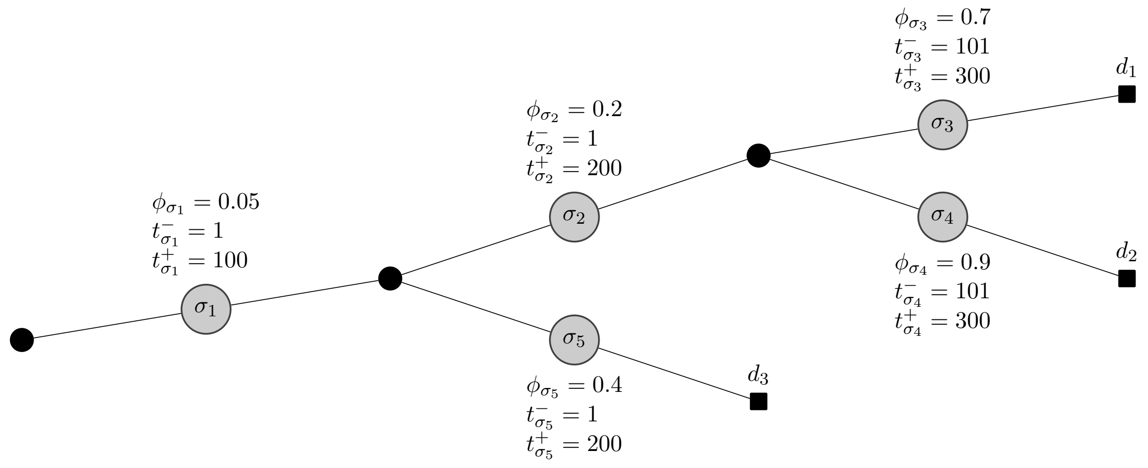


Figure 4: Simple scenario for the S&T problem with one observer.

the target has not been discovered earlier is $P_{\sigma_5 *} = \phi_{\sigma_5} P_\emptyset(\gamma_3) = \frac{2}{15}$. The total probability of finding the target after the execution of σ_5 is therefore $P(\sigma_5) = \frac{2}{15}$. For the next iteration, we update the probabilities that the target is following each path. For the path γ_3, which is compatible with σ_5, the probability decreases: $P_{\sigma_5}(\gamma_3) = \frac{P_\emptyset(\gamma_3)(1-\phi_{\sigma_5})}{1-P_{\sigma_5 *}} = \frac{3}{13}$, while for the other two paths, the probability increases: $P_{\sigma_5}(\gamma_1) = P_{\sigma_5}(\gamma_2) = \frac{P_\emptyset(\gamma_1)}{1-P_{\sigma_5 *}} = \frac{5}{13}$. We can now compute the probability of finding the target when executing σ_4 after σ_5: $P_{\sigma_5,\sigma_4 *} = \phi_{\sigma_4} P_{\sigma_5}(\gamma_2) = \frac{9}{26}$ and $P(\sigma_5, \sigma_4) = P(\sigma_5) + P_{\sigma_5,\sigma_4 *} \cdot (1 - P(\sigma_5)) = \frac{13}{30}$. Similarly, for the final step, we update the probability for each path $P_{\sigma_5,\sigma_4}(\gamma_1) = \frac{10}{17}$, $P_{\sigma_5,\sigma_4}(\gamma_2) = \frac{1}{17}$ and $P_{\sigma_5,\sigma_4}(\gamma_3) = \frac{6}{17}$. The probability of finding the target when executing σ_3 after σ_5 and σ_4 is $P_{\sigma_5,\sigma_4,\sigma_3 *} = \phi_{\sigma_3} P_{\sigma_5,\sigma_4}(\gamma_1) = \frac{7}{17}$ and the total probability is $P(\sigma_5, \sigma_4, \sigma_3) = P(\sigma_5, \sigma_4) + P_{\sigma_5,\sigma_4,\sigma_3 *} \cdot (1 - P(\sigma_5, \sigma_4)) = \frac{2}{3}$.

3. Properties of the Objective Function

In this section, we analyze the properties of the objective function P. We first observe that, despite having a recursive definition, the total probability of finding the target is defined over sets Σ of search patterns (i.e., it is a set function) and so does not depend on the order of the elements that appear in Σ. This property comes from our definition of total probability, which abstracts the notion of time and depends only on whether or not we executed a particular combination of search patterns. In fact, the dependency on time of the candidate search patterns is pre-computed thanks to the notion of time window attached to each pattern so that, since a pattern is located both in space and in time, the probabilities for the target position can be assumed to depend only on the search patterns executed so far (and not on the exact time the pattern is executed). To show this property we can simply expand the recursive definition of total probability and show that all the terms are commutative.

Proposition 3.1. The total probability $P(\Sigma)$ is a set function.

Proof. To see this more directly, given a set $\Sigma = \{\sigma_1, \dots, \sigma_n\}$, it is sufficient to write the value of $P(\Sigma)$ as follows:

$$\begin{aligned}
P(\{\sigma_1, \dots, \sigma_n\}) =& P(\{\sigma_1, \dots, \sigma_{n-1}\}) + P_{\{\sigma_1,\dots,\sigma_n\}*}(1 - P(\{\sigma_1, \dots, \sigma_{n-1}\})) \\
=& P(\{\sigma_1, \dots, \sigma_{n-1}\}) + P_{\{\sigma_1,\dots,\sigma_n\}*} \prod_{j=1}^{n-1} (1 - P_{\{\sigma_1,\dots,\sigma_j\}*}) \\
=& \sum_{i=1}^{n} P(\{\sigma_1, \dots, \sigma_i\}) \prod_{j=1}^{n-1} (1 - P_{\{\sigma_1,\dots,\sigma_j\}*}) \\
=& \sum_{i=1}^{n} \phi_{\sigma_i} \sum_{\gamma\in\Gamma} P_{\{\sigma_1,\dots,\sigma_{i-1}\}}(\gamma) \mathbb{1}_{\Gamma_{\sigma_i}} \prod_{j=1}^{n-1} (1 - P_{\{\sigma_1,\dots,\sigma_j\}*}) \\
=& \sum_{\gamma\in\Gamma} P_{\emptyset}(\gamma) \sum_{i=1}^{n} \phi_{\sigma_i} \mathbb{1}_{\Gamma_{\sigma_i}} \prod_{j=1}^{i-1} (1 - \phi_j \mathbb{1}_{\Gamma_{\sigma_j}}) \\
=& \sum_{\gamma\in\Gamma} P_{\emptyset}(\gamma) \left(\sum_{s\in 2^{\Sigma}} \prod_{\sigma_i \in s} (-1)^{2|s|+1} \phi_{\sigma_i} \mathbb{1}_{\Gamma_{\sigma_i}} \right)
\end{aligned} \tag{8}$$

Clearly, Equation (8) does not depend on the order of the elements in Σ. □

Since performing more search patterns can only increase our chance to observe the target, we show that our objective function is non-decreasing.

Proposition 3.2. The total probability function $P(\Sigma)$ is a non-decreasing function.

Proof. Let $\Sigma \subset \widetilde{\mathcal{C}}, \sigma \in \widetilde{\mathcal{C}}$ respectively be a set of search patterns and a single search pattern, with $\sigma \notin \Sigma$. We show that:

$$P(\Sigma \cup \{\sigma\}) - P(\Sigma) = P_{\Sigma\cup\{\sigma\}*}(1 - P(\Sigma)) \geq 0$$

This is satisfied because $P_{\Sigma\cup\{\sigma\}*} \geq 0$, $P(\Sigma) \leq 1$. □

We now prove that the objective function is submodular. In mathematical optimization, problems characterized by monotonic *submodular set functions* have been extensively studied and it has been proven that greedy algorithms are guaranteed to produce solutions close to optimality (Nemhauser & Wolsey, 1978).

Let N be a finite set and $f : 2^N \to \mathbb{R}$ be a set function that assigns each subset $H \subseteq N$ a value $f(H)$. The function f is called *submodular* if for every $X, Y \subseteq N$, with $X \subseteq Y$ and every $x \in N \setminus Y$, we have that $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$. Intuitively, if f is a utility function, submodularity captures the concept that the *marginal* utility of adding x to a set X is at least as high as the utility of adding it to a superset Y.

Proposition 3.3. Given a set $\Sigma \subseteq \widetilde{\mathcal{C}}$, the total probability function $P(\Sigma)$ is submodular.

Proof. Let $\Sigma \subseteq \widetilde{\mathcal{C}}, \sigma \in \widetilde{\mathcal{C}}$ respectively be a set of search patterns and a single search pattern, with $\sigma \notin \Sigma$. Consider $\Psi \subseteq \Sigma$ such that $\Psi = \{\sigma_{f(1)}, \dots, \sigma_{f(k)}\}$, with $k \leq n$ and $f : \mathbb{N} \to \mathbb{N}$ a monotonic function with $f(k) \leq n$, then:

$$P(\Psi \cup \{\sigma\}) - P(\Psi) \geq P(\Sigma \cup \{\sigma\}) - P(\Sigma) \tag{9}$$

Considering the r.h.s. of the inequality and using Equation (1) first, then Equation (32) and finally Equations (6) and (7), we obtain:

$$
\begin{aligned}
&P(\Sigma \cup \{\sigma\}) - P(\Sigma) \\
&= P_{\Sigma\cup\{\sigma\}*}(1 - P(\Sigma)) \\
&= \phi_\sigma \sum_{\gamma\in\Gamma_\sigma} P_\Sigma(\gamma)(1 - P(\Sigma)) \\
&= \phi_\sigma \cdot \sum_{\gamma\in\Gamma_\sigma} \frac{P_\emptyset(\gamma)\cdot \prod_{i=1}^{n}(1-\phi_{\sigma_i}\cdot \mathbb{1}_{\Gamma_{\sigma_i}}(\gamma))}{\prod_{i=1}^{n}(1-P_{\{\sigma_1,\cdots,\sigma_i\}*})}(1-P(\emptyset))\prod_{i=1}^{n}(1-P_{\{\sigma_1,\cdots,\sigma_i\}*}) \\
&= \phi_\sigma(1-P(\emptyset))\sum_{\gamma\in\Gamma_\sigma}\left[P_\emptyset(\gamma)\cdot\prod_{i=1}^{n}(1-\phi_{\sigma_i}\cdot \mathbb{1}_{\Gamma_{\sigma_i}}(\gamma))\right] \qquad (10)
\end{aligned}
$$

We now consider the l.h.s. of the inequality. We can apply the same line of reasoning applied to the r.h.s:

$$
\begin{aligned}
&P(\Psi \cup \{\sigma\}) - P(\Psi) \\
&= \phi_\sigma(1-P(\emptyset))\sum_{\gamma\in\Gamma_\sigma}\left[P_\emptyset(\gamma)\cdot\prod_{i=1}^{k}(1-\phi_{\sigma_{f(i)}}\cdot \mathbb{1}_{\Gamma_{\sigma_{f(i)}}}(\gamma))\right] \qquad (11)
\end{aligned}
$$

Let us now compare Equations (10) and (11). First, we observe that every term in these equations is non-negative. The only difference is the presence in Equation (10) of the factors $\prod_{i=1}^{n}(1-\phi_{\sigma_i}\cdot \mathbb{1}_{\Gamma_{\sigma_i}}(\gamma))$, such that $i \neq f(j), \forall j = 1,\ldots,k$. Since this term is less or equal to one, the result follows. $\square$

4. Solution Approaches

In this section, we describe three different solution approaches for the search problem described in Section 2. In particular, given the pattern candidates $\widetilde{\mathcal{C}}$, the set of possible paths Γ, the observers $\mathcal{O}$ with their associated distance matrices $\widetilde{D}^o$ for each $o \in \mathcal{O}$, we want to find a set Σ of search patterns such that Σ is *executable* and maximizes the objective function P. Note that for all approaches, the sequence is calculated offline: during the computation of the solution, the solver adds a new pattern to the plan under the assumption that the previous pattern has failed. If a pattern had been successful, the sequence becomes irrelevant as the UAV switches back to tracking. Failed patterns give the planner (negative) information about the position of the target, which is reflected in the updated probabilities. At execution time, the search patterns in the sequence are executed by the UAVs until the target is found.

Given the submodularity of the objective function, we first explore a greedy algorithm that incrementally builds the set Σ by choosing, at each step, the search pattern that yields the highest increment for the objective function and by assigning it to an observer that is free to execute it. We then consider two theoretically complete approaches by modelling the search problem, respectively, as a constraint program and an AI planning task. For both these approaches, we consider how to take advantage of the greedy solution to improve the performance of the solvers.

4.1 Greedy Algorithms

Greedy algorithms perform very well when used to solve problems that involve the maximization of a submodular non-decreasing function (Lin & Bilmes, 2010; Krause & Guestrin, 2011; Alon, Gamzu, & Tennenholtz, 2012; Jawaid & Smith, 2015; Parambath, Vijayakumar, & Chawla, 2018). In fact, such greedy algorithms have a theoretical guarantee to find $(1 - 1/e)$-approximations. In our problem, however, this guarantee is not ensured because, although the objective function is non-decreasing and submodular, a solution must satisfy the temporal constraints associated with the patterns. In our experimental evaluation, we study the performance of the greedy algorithm and analyze how imposing executability impacts its performance.

Suppose that the set of search pattern candidates $\widetilde{\mathcal{C}}$, the set of observers $\mathcal{O}$, the set of possible destinations Γ, and the set of distance matrices $D = \{\widetilde{D}^o, \forall o \in \mathcal{O}\}$ are given. The greedy procedure, outlined in Algorithm 1, proceeds iteratively to construct a set of patterns according to the recursive structure of Equation (1). The algorithm starts from an empty set (Algorithm 1: line 2) and iteratively adds a search pattern to the set if it can find an observer that can execute the pattern considering the temporal constraints (Algorithm 1: lines 7-14). The assignment algorithm (Algorithm 2) runs through all the possible observers and, for each observer (Algorithm 2: line 2), iterates over the sequence of the search patterns that are already assigned to it in the time window associated with the new pattern (Algorithm 2: line 3). The algorithm tries to insert the new pattern between each consecutive pair of patterns by checking the consistency of the temporal constraints imposed by the new sequence (Algorithm 2: lines 5-6). These temporal constraints can be checked in linear time with respect to the number of elements in the sequence, as shown in the procedure CheckConsistency. For every element s_i of a sequence $S = (s_1, \ldots, s_i, \ldots, s_n)$, the procedure assigns the exact start time $\tau(s_i)$ to s_i by taking the maximum time between the sum of the end time of s_{i-1} and the travel time between s_{i-1} and s_i and the minimum time $t^-_{s_i}$ (Algorithm 2: lines 9-12). If the calculated time $\tau(s_i)$ is greater than $t^+_{s_i}$, the pattern cannot be executed and the sequence is marked as not-executable (Algorithm 2: line 11). The greedy algorithm returns the set of patterns Σ to be executed and their associated observers. If we group the patterns associated with each observer into sets, they clearly form a partition of Σ.

We also devise an alternative version of the assignment algorithm, which we call Right Assignment Algorithm. For each observer, we add the new candidate search pattern at the end of sequence built so far instead of iterating over all possible positions (Algorithm 2: line 3) in the sequence. The greedy algorithm with this variation of the assignment problem can be also interpreted as an *online* algorithm, where the search pattern with the highest reward is assigned to one of the available observers.

To see the difference between the two assignment algorithms, consider the problem in Example 2.1. Let us start with Algorithm 1 in combination with the Right Assignment

Algorithm 1 Greedy Algorithm

```
1: procedure Greedy($\widetilde{\mathcal{C}}, \mathcal{O}, \Gamma, D$)
2:     $S(o) \leftarrow \emptyset \qquad \forall o \in \mathcal{O}$
3:     return UpdateSequence($\widetilde{\mathcal{C}}, \mathcal{O}, \Gamma, D, \emptyset, S$)
4: procedure UpdateSequence($\widetilde{\mathcal{C}}, \mathcal{O}, \Gamma, D, \Sigma, S$)
5:     for $i = 0, ..., \max_{\sigma \in \mathcal{C}}(t^+_\sigma + d_\sigma) / \min_{\sigma \in \mathcal{C}}(d_\sigma)|\mathcal{O}|$ do
6:         $\delta \leftarrow 0$
7:         for $\sigma \in \widetilde{\mathcal{C}} \setminus \Sigma$ do
8:             $\Delta \leftarrow P(\Sigma \cup \{\sigma\}) - P(\Sigma)$
9:             if $\Delta \geq \delta$ then
10:                $(S^*, o^*) \leftarrow$ Assignment($\mathcal{O}, \widetilde{\mathcal{C}}, D, \sigma, S$)
11:                if $(S^*, o^*) \neq (\emptyset, \emptyset)$ then
12:                    $\sigma^* \leftarrow \sigma$
13:                    $\delta \leftarrow \Delta$
14:                    $(S^*, o^*) \leftarrow (S, o)$
15:        if $\delta \leq 0$ then return $\Sigma, S$
16:        else
17:            $\Sigma \leftarrow \Sigma \cup \{\sigma_{best}\}$
18:            $S(o^*) \leftarrow S^*$
       return $\Sigma, S$
```

Algorithm 2 Assignment Algorithm

```
1: procedure Assignment($\mathcal{O}, \widetilde{\mathcal{C}}, D, \sigma, S$)
2:     for $o \in \mathcal{O}$ do
3:         for $i = 1, \ldots, |S^o|$ do
4:             $S'^o \leftarrow (s_1, \ldots, s_i, \sigma, s_{i+1}, \ldots s_{|S^o|})$
5:             if CheckConsistency($S'^o, \widetilde{\mathcal{C}}, D^o$) then
6:                 $(S^*, o^*) \leftarrow (\sigma, S'^o, o)$
7:                 return $(S^*, o^*)$
       return $(S^*, o^*)$
8: procedure CheckConsistency($S, \widetilde{\mathcal{C}}, D^o$)
9:     $t \leftarrow \max(D_{s_0,s_1}, t^-_{s_1})$
10:    for $i = 1 \ldots |S| - 1$ do
11:        if $t > t^+_{s_i}$ then return $False$
12:        $t \leftarrow \max(t + d_{s_i} + D_{s_i,s_{i+1}}, t^-_{s_{i+1}})$
       return $t \leq t^+_{s_{|S|}}$
```

Algorithm that adds search patterns at the end of the sequence only. The search pattern returning the highest probability of finding the target is σ_4, which becomes the first element of the sequence. The observer will then execute σ_3 (the second best pattern), after which, due to the temporal constraints, it cannot execute any other patterns. The total probability of this solution is $P(\sigma_4, \sigma_3) = \frac{8}{15}$. However, we can improve on this solution by using the standard Assignment Algorithm (Algorithm 2). After choosing σ_4 and σ_3, Algorithm 2 selects search pattern σ_5 and positions it at the beginning of the sequence, when it can be executed. In this way, it finds the optimal solution for this problem.

We now show, with a counterexample, that the quality of the solutions provided by the greedy algorithm can be lower than $(1 - 1/e)$ of the optimal solution.

Example 4.1. Consider a set of candidate search patterns $\mathcal{C} = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$, a set of paths $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$ and a single observer. The parameters associated with each search pattern are reported in Table 1, where $D^o(\sigma, \rho)$ is an element of the symmetric distance matrix and ρ is either 0 or another search pattern different than σ. We assume a uniform initial probability distribution for the every path.

σ	Γ_σ	ϕ_σ	$[t^-_\sigma, t^+_\sigma]$	d_σ	$D^o(\sigma,\rho)$	$D^o(\sigma,\sigma)$
σ_1	$\{\gamma_1\}$	1	$[10, 11]$	2	10	2
σ_2	$\{\gamma_2\}$	0.8	$[1, 2]$	2	1	2
σ_3	$\{\gamma_3\}$	0.8	$[4, 5]$	2	1	2
σ_4	$\{\gamma_4\}$	0.8	$[7, 8]$	2	1	2

Table 1: Search pattern candidates.

In Example 4.1, search pattern σ_1 is associated with the highest detection probability, but it is located further away from the other search patterns. The optimal solution is the sequence $\Sigma^* = (\sigma_2, \sigma_3, \sigma_4)$, which yields a total probability of $P(\Sigma^*) = \frac{3}{5}$. The greedy algorithm is initially agnostic to the temporal constraints and selects σ_1, which will be execution at time 10. No further search patterns can be inserted in the sequence, as none of them can be executed before nor after σ_1. The solution $\Sigma^G = \{\sigma_1\}$ found by the greedy algorithm has a total probability of $P(\Sigma^G) = \frac{1}{4}$. Clearly, $P(\Sigma^G) < (1 - 1/e)P(\Sigma^*)$.

4.2 A Constraint Programming Approach

Constraint Programming (CP) is a technique to solve combinatorial problems that involve decision variables subject to a set of constraints. Constraint solvers explore the solution space systematically by interleaving search and inference until a solution is found. CP has been successfully applied to many different domains to solve discrete optimization and scheduling problems (Weil, Heus, Francois, & Poujade, 1995; Beck, Davenport, Davis, & Fox, 1998; Baptiste, Le Pape, & Nuijten, 2001; Harjunkoski & Grossmann, 2002; Rodriguez, 2007), as well as robot task planning problems (Booth, Nejat, & Beck, 2016a; Booth, Tran, Nejat, & Beck, 2016b; Tran, Vaquero, Nejat, & Beck, 2017). Target search lends itself well to a CP formulation, given the combinatorial nature of the problem and the presence of the temporal constraints.

A CP model for problems with a single observer was proposed by Bernardini et al. (2017b). The model discretizes time into a set of time points and uses time-indexed binary variables for each search pattern indicating whether the search pattern is being executed at a time point. Bernardini et al. (2017b) use a coarse time discretization because, if the discretization is too fine, the number of variables becomes very large, and the solver struggles to find a solution. A coarse discretization, however, could make the representation of the temporal constraints inaccurate, possibly resulting in some of the search patterns in the solution not being executable.

In this work, we propose a new CP model for the target search problem that overcomes the limits of the previous model. Our new model is able to handle multiple observers and exploits *optional interval variables* to represent the execution of search patterns, improving accuracy and scalability. We start in Section 4.2.1 with a generalization of the model presented in the work by Bernardini et al. (2017b) to multiple observers, which will be useful for an experimental comparison between the different models. We then describe our new model in Sections 4.2.2.

4.2.1 Extended Time-discretized CP Model

We extend the CP model proposed by Bernardini et al. (2017b) to multiple observers, as shown in Figure 5. The model requires the discretization of time into a set of τ time points $\mathcal{T} = \{t_0, \ldots, t_\tau\}$, where $t_\tau = \lceil \max_{\sigma \in \mathcal{C}}(t^+_\sigma + d_\sigma)/\Delta \rceil$ and Δ is the discretization granularity. A binary variable $z_{\sigma,t,o} = \{0, 1\}$ for every $\sigma \in \mathcal{C} \cup \{\sigma_0\}$, where σ_0 is a dummy node representing the initial position of the observers, $o \in \mathcal{O}$ and $t \in \mathcal{T}$ indicates if a pattern σ starts at time point t. In addition, the continuous variables P_t, $\forall t \in \mathcal{T}$ are used to indicate the total probability of finding the target at time point t; P^*_t, $\forall i \in \mathcal{T}$ indicate the probability of finding the target when executing a search pattern at time point t provided that the target has not been discovered earlier, $P_{t,\gamma}$ $\forall \gamma \in \Gamma, \forall t \in \mathcal{T}$ to indicate the probability that the target is following path γ at time point t.

The model finds the values of $z_{\sigma,t,o}$, $\forall\ \sigma \in \mathcal{C}$, $\forall t \in \mathcal{T}$, $\forall o \in \mathcal{O}$ such that the weighted sum of the total probability in the last time point t_τ is maximized and is subject to the following constraints (see Figure 5): Constraint (13) ensures that only one pattern at the time can be executed by each observer, while Constraint (14) indicates that a pattern can be executed only within its time window. Two patterns σ and ρ can be performed one after the other only if the sum of the time needed to execute σ and the time needed to reach ρ has elapsed, as shown by Constraint (15). Constraints (16)-(18) maintain the total probability in order to calculate the objective function. It should be noted that if no search pattern is started at a given time point, the total probability does not change. Constraints (19)-(22) represent the initial state.

This model does not require the explicit representation of possible repeated search patterns: each element in $\mathcal{C}$ can be potentially executed multiple times by every observer, provided that the temporal Constraints (14)-(15) are satisfied.

4.2.2 CP Model based on Optional Interval Variables

In our new model, to limit the number of variables necessary to represent the search problem, we use optional interval variables, which allow us to avoid time-indexing (Laborie & Rogerie, 2008; Laborie, 2009). Optional interval variables have been successfully used in many scheduling applications (Booth et al., 2016a; Tran et al., 2017; Laborie & Messaoudi, 2017) to represent tasks. They are variables that encode three quantities: whether or not an activity is executed, the start time and the end time of the activity. More formally, optional interval variables have domains of the form $\{\perp\} \cup \{[s, e)$ s.t. $s, e \in \mathbb{Z}, s < e\}$, where $\perp$ indicates that the variable is not present in the solution, while s and e are the start and the end points of the interval. Associated with these variables, there are several functions and constraints, for which efficient filtering algorithms have been developed. Constraints are

$$\max P_{t_\tau} \tag{12}$$

$$\sum_{\sigma\in\mathcal{C}} z_{\sigma,t,o} \leq 1 \qquad \forall t \in \mathcal{T}, \forall o \in \mathcal{O} \tag{13}$$

$$z_{\sigma,t,o} = 0 \qquad \forall\sigma \in \mathcal{C}, \forall o \in \mathcal{O}, \forall t \in \mathcal{T} | t < t_\sigma^- \land t \geq t_\sigma^+ \tag{14}$$

$$z_{\sigma,t,o} + \sum_{i=t}^{t+(d_\sigma + D_{\sigma,\rho}^o)/\Delta} z_{\rho,i,o} \leq 1 \qquad \forall\sigma,\rho \in \mathcal{C}, \forall o \in \mathcal{O}, \forall t \in \mathcal{T} \tag{15}$$

$$P_t^* = \sum_{\sigma\in\mathcal{C}}\sum_{\gamma\in\Gamma}\sum_{o\in\mathcal{O}} \left(P_{\gamma,t-1}\phi_\sigma \mathbb{1}_{\Gamma_\sigma}\right) z_{\sigma,t,o} \qquad \forall t \in \mathcal{T}\setminus\{t_0\} \tag{16}$$

$$P_{\gamma,t} = P_{\gamma,t-1}\left(1 - \sum_{\sigma\in\mathcal{C}}\sum_{o\in\mathcal{O}}\left(1 - \frac{1-\phi_\sigma \mathbb{1}_{\Gamma_{\sigma_j}}}{1-P_t^*}\right) z_{\sigma,t,o}\right) \qquad \forall t \in \mathcal{T}\setminus\{t_0\}, \gamma\in\Gamma \tag{17}$$

$$P_t = P_{t-1} + \sum_{\sigma\in\mathcal{C}} P_t^* \left(1 - P_{t-1}\right) z_{\sigma,t,o} \qquad \forall t \in \mathcal{T}\setminus\{t_0\} \tag{18}$$

$$z_{\sigma_0,0,o} = 1 \qquad \forall o \in \mathcal{O} \tag{19}$$

$$P_0 = 0 \tag{20}$$

$$P_{0,\gamma} = PD(\gamma) \qquad \forall\gamma\in\Gamma \tag{21}$$

$$P_0^* = 0 \tag{22}$$

Figure 5: Time-indexed CP model for the target search problem extending the model of Bernardini et al. (2017b) to multiple observers.

divided into logical, temporal and hybrid constraints. Logical constraints express conditions on the executions of interval variables, while temporal constraints involve the start and the end time of the variables. Hybrid constraints are constraints that combine both the logical and the temporal aspect of the variables. An example of a hybrid constraint is `noOverlap`$(x_0, \dots, x_n, D)$. This constraint is set over the interval variables $\{x_0, \dots, x_n\}$ and a distance matrix D that enforces that the present intervals are pairwise non-overlapping and that a minimal distance $D_{x_i,x_{i+1}}$ between the end and the start of two consecutive present interval variables, x_i, x_{i+1}, is respected. Figure 6 shows an example of a `noOverlap` constraint imposed to three interval variables x_1, x_2, x_3, which distance matrix $D_{x_i,x_j} = 1, \forall i,j = 1,2,3$.

A key observation in our CP formulation is that the total probability does not depend on the order of the execution of the search patterns, and therefore, to update its value, we can choose an arbitrary order and use the recursive definition in Equation (1).

Given the problem defined in Section 2 and a set of patterns $\widetilde{\mathcal{C}} \cup \{\sigma_0\}$, where σ_0 is a dummy node representing the initial position of the observers, we create an arbitrary sequence Σ containing $|\mathcal{O}|$ copies of each search pattern and beginning with σ_0. We associate a search pattern $\sigma_i \in \widetilde{\mathcal{C}}$ and an observer $o_i \in \mathcal{O}$ to every element $i \in \Sigma$. The decision variables are: (i) *optional interval variables* $t_{\sigma,o}$, present if the search pattern $\sigma \in \widetilde{\mathcal{C}} \cup \{\sigma_0\}$

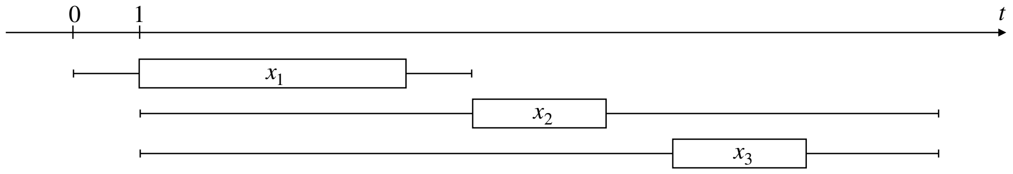


Figure 6: `noOverlap` constraint on three interval variables x_1, x_2, x_3. Variable x_1 can start from 0 and end before 6 and has a duration of 4 while x_2 and x_3 can start after 1, end before 13 and have duration 2. The constraint enforces that none of the intervals overlap in time. By convention, interval variables are half-open: closed at their start times, open at their end times.

is executed by the observer $o \in \mathcal{O}$; (ii) continuous variables P_i, $\forall i \in \Sigma$, which indicate the total probability of finding the target after the execution of the i^{th} element of Σ; (iii) P_i^*, $\forall i \in \Sigma$, which indicate the probability of finding the target by executing the pattern at step i provided that the target has not been discovered earlier; and (iv) $P_{i,\gamma}$ $\forall \gamma \in \Gamma, \forall i \in \Sigma$, which indicate the probability that the target is following path γ after the i^{th} element of Σ.

Besides the `noOverlap` constraint, we use the following functions: `Length`(x) is a function returning the length of the interval x, if the interval is present; `Pres`(x) is a function that returns a boolean value that indicates whether the variable x is present or not.

The CP model is defined in Figure 7. Constraint (24) models the travel time between two consecutive search patterns executed by the same observer, while Constraint (25) sets the length of every search pattern. Constraints (26)-(28) update the total probabilities recursively, Constraint (29) sets the initial position of the observers and Constraints (20)-(22) set the initial values of the variables representing the probabilities.

4.2.3 Exploiting the Submodularity of the Objective Function

To exploit submodularity, we warm start the CP solver with a greedy solution. Warm start is a procedure that exploits a known solution by using it as a starting point of another algorithm. The warm-start solution can be used by a CP solver to impose a lower bound on the quality of the solution, allowing additional inference and heuristic guidance of the search (Beck, 2007). Details on how the warm-start is used inside CP solvers largely depend on the implementation of the solver and they are opaque to the final user (Laborie, Refalo, & Shaw, 2013).

4.3 A Planning Approach

In formulating and solving the search problem described in Section 2 as a planning task, we build on the planning model for a single observer presented in the work by Bernardini et al. (2017b) and extend it to handle a team of UAVs. This extension is straightforward, essentially involving the addition of a parameter to each action in the domain that indicates which UAV will execute the action and the enumeration of the available UAVs in the initial

$$
\begin{aligned}
&\max P_{|\Sigma|} && && (23)\\
&\text{s.t.}\,\texttt{noOverlap}(t_{\sigma_0,o}, t_{\sigma_1,o}, ..., t_{\sigma_{|\widetilde{\mathcal{C}}|},o}; D^o) && \forall o \in \mathcal{O} && (24)\\
&\quad \texttt{Length}(t_{\sigma,o}) = d_\sigma && \forall \sigma \in \widetilde{\mathcal{C}}, \forall o \in \mathcal{O} && (25)\\
&\quad P_i^* = \sum_{\gamma\in\Gamma} \left(P_{i,\gamma}\phi_\sigma \mathbb{1}_{\Gamma_{\sigma_i}}\right) && \forall i \in \Sigma \setminus \{\sigma_0\} && (26)\\
&\quad P_{i,\gamma} = P_{i-1,\gamma}\left(1 - \left(1 - \frac{1-\phi_\sigma \mathbb{1}_{\Gamma_{\sigma_i}}}{1-P_i^*}\right)\texttt{Pres}(t_{\sigma_i,o_i})\right) && && \\
& && \forall i \in \Sigma \setminus \{\sigma_0\}, \forall \gamma \in \Gamma && (27)\\
&\quad P_i = P_{i-1} + P_i^*\left(1 - P_{i-1}\right)\texttt{Pres}(t_{\sigma_i,o_i}) && \forall i \in \Sigma \setminus \{\sigma_0\} && (28)\\
&\quad \texttt{Pres}(t_{\sigma_0,o}) = 1 && \forall o \in \mathcal{O} && (29)\\
&\quad \text{Constraints (20)-(22)} && &&
\end{aligned}
$$

Figure 7: CP Model based on optional interval variables for the target search problem.

state specification. All the other components of the planning system remain unaltered, demonstrating the benefits of using general-purpose solvers: when the task at hand changes, only the model needs to be updated, not the algorithms. More significantly, based on our recognition of the submodularity of the objective function, we integrate the greedy heuristic as a planning heuristic.

The PDDL (Fox & Long, 2003) model that we use presents a set of objects that corresponds to the main symbols in the mathematical formulation in Section 2: the search patterns, the paths that the target might follow to reach its destination, the relevant waypoints (start and end points of the patterns as well as the last known position of the target) and the UAVs. We use functions to keep track of the total probability and the path probabilities, both at the current step (respectively, $P(\Sigma)$ and $P_\Sigma(\gamma)$) and when the current plan is concatenated with a new pattern (respectively, $P(\Sigma \cup \sigma)$ and $P_{\Sigma\cup\{\sigma\}}(\gamma)$). We have a *fly* action, which corresponds to a UAV going from one point in space to another and one action for each search pattern (*doPattern*). All these actions have a parameter that indicates which UAV is executing them. The conditions of the actions that represent search patterns check that the UAV is ready at the entry point of the pattern to execute it and the pattern is active, i.e., the associated time window has started. The effects move the UAV from the entry point to the exit point of the pattern and, crucially, update the functions corresponding to probabilities according to Equations (1)-(4). The full PDDL domain is presented in Appendix C.

The initial state of the planning task contains information regarding all the available UAVs, the relevant points in space, the paths that the target might follow to arrive at its destination and the activation windows of each pattern, which are specified by using *timed initial literals* (TIL). The objective of the planning task is to maximize the objective function P, which is expressed in the metric of the planning task.

We use the planner popf-tif (Piacentini, Alimisis, Fox, & Long, 2015) to build plans for the UAVs. popf-tif is based on the partial order temporal planner popf2 (Coles et al., 2010) and uses a cost-improving search: it finds the first plan and then improves on the first solution while time is available. Since the PDDL model for the search problem involves non-linear mathematical calculations, we combine popf-tif with an external solver to calculate Equations (1)-(4) at each iteration based on the approach described by Bernardini et al. (2017a).

4.3.1 Exploiting the Submodularity of the Objective Function

The planner popf-tif relies on a greedy best-first search (GBFS) algorithm (Doran & Michie, 1966), where the heuristic is calculated as the length of the relaxed plan built from the Temporal Relaxed Planning Graph (TRPG) (Smith & Weld, 1997; Coles, Fox, Halsey, Long, & Smith, 2009; Coles, Coles, Fox, & Long, 2009; Coles et al., 2010). This heuristic is not particularly effective in our domain. In fact, until a first solution is found, the heuristic function is completely insensitive to the objective function of the problem, as the TRPG tries to find the shortest relaxed plan that satisfies the goal conditions, which in our case corresponds to the addition of a single search pattern in the plan. After a feasible solution is found, a further goal condition is imposed requiring a new goal state to have a metric function strictly greater than the best metric found so far. However, the heuristic function still favours goal states that are achieved by fewer steps, rather than goal states with a higher objective function.

To improve the performance of the planner, we exploit the submodularity of the objective function and propose a domain-dependent heuristic based on the greedy solution that takes into account the objective function of the problem.

Starting from the partial plan π necessary to achieve a state, Algorithm 3 calculates the remaining part of the plan $\hat{\pi}$, based on the sequence of search patterns that the greedy algorithm would produce (Algorithm 1) and returns the heuristic value of the state and the number of actions in $\hat{\pi}$. We use the symbols $\vdash$ and $\dashv$ to indicate the start and the end of a durative action, respectively. The heuristic function of the state is calculated as the difference between 1 (the maximum theoretical value of a solution) and the objective value of the partial plan $\pi \cup \hat{\pi}$. We use the difference because our greedy best-first-search algorithm expands states with lower heuristic value, but we have a maximization problem. The number of actions in $\hat{\pi}$ is used to break ties when states have the same heuristic value.

The heuristic is not admissible, hence the optimality of the solutions can be guaranteed only if the search explores the entire space. However, since the heuristic function is based on feasible sequences, the first plan found by the algorithm, in which no more candidates can be inserted, is guaranteed to be at least as good as the solution found by the greedy algorithm.

More formally, let $\mathcal{Z}$ be a finite set, $2^{(\mathcal{Z})}$ the set of all possible sequences constructed over $\mathcal{Z}$ and C a set of constraints over the elements in $\mathcal{Z}$, and $f : 2^{(\mathcal{Z})} \rightarrow \mathbb{R}$ a sequence function that we want optimize. We call a sequence $S \in 2^{(\mathcal{Z})}$ a *maximal sequence* if $\nexists z \in \mathcal{Z}$ such that $S \cup \{z\}$ satisfies all the constraints in C. We define the state transition graph $\mathcal{G}(\mathcal{Z}, C) = \langle \mathcal{S}, \mathcal{T} \rangle$ where the set of states $\mathcal{S}$ is the set of all the feasible sequences of $\mathcal{Z}$ (i.e. the sequences in $2^{(\mathcal{Z})}$ that satisfy C) and the transitions $\mathcal{T}$ are the concatenation operations

Algorithm 3 Greedy Algorithm-Based Heuristic

```
1: procedure HEURISTIC($\mathcal{C}, \mathcal{O}, \Gamma, D, \pi$)
2:     $\Sigma \leftarrow \emptyset$
3:     $S(o) \leftarrow \emptyset \quad \forall o \in \mathcal{O}$
4:     for $a \in \pi$ do
5:         if $a$ is (fly $o$ $\sigma_{from}$ $\sigma_{to}$) then
6:             $\Sigma \leftarrow \Sigma \cup \sigma_{to}$
7:             $S(o) \leftarrow S(o) \cup \sigma_{to}$
8:     $\hat{\Sigma}, S \leftarrow$ UpdateSequence($\mathcal{C}, \mathcal{O}, \Gamma, D, \Sigma, S$)
9:     $t \leftarrow 0$
10:    for $\sigma \in \hat{\Sigma}$ do
11:        $o \leftarrow o$ s.t. $\sigma \in S(o)$
12:        if (fly $o$ $\sigma^o$ $\sigma$) not in $\pi$ then $\hat{\pi} \leftarrow \hat{\pi} \cup$(fly $o$ $\sigma^o$ $\sigma$)
13:        if (doPattern $o$ $\sigma$)$\vdash$ not in $\pi$ then $\hat{\pi} \leftarrow \hat{\pi} \cup$(doPattern $o$ $\sigma$)$\vdash$
14:        if (doPattern $o$ $\sigma$)$\dashv$ not in $\pi$ then $\hat{\pi} \leftarrow \hat{\pi} \cup$(doPattern $o$ $\sigma$)$\dashv$
15:        if $t^-_\sigma > t$ then $t \leftarrow t^-_\sigma$
16:        $\sigma^o \leftarrow \sigma$
17:    for $TIL \leq t$ do
18:        if $TIL$ not in $\pi$ then $\hat{\pi} \leftarrow \hat{\pi} \cup TIL$
19:    return $1 - P(\hat{\Sigma}), |\hat{\pi}|$
```

of one element $z \in \mathcal{Z}$ to a sequence $S \in 2^{(\mathcal{Z})}$. We denote with $\mathcal{A}_{\mathcal{Z},C} : 2^{(\mathcal{Z})} \rightarrow 2^{(\mathcal{Z})}$ an algorithm that produces a maximal sequence $\bar{S}$ from a subsequence $S \in 2^{(\mathcal{Z})}$, such that $\bar{S} = S \cup S'$. In addition, we required that $\mathcal{A}_{\mathcal{Z},C}$ produces a consistent solution, i.e. $f(A_{\mathcal{Z},\mathcal{C}}(S \cup \{\sigma\}) \leq f(A_{\mathcal{Z},\mathcal{C}}(S))$, for σ s.t. $\sigma \in S'$ where $A_{\mathcal{Z},\mathcal{C}}(S) = S \cup S'$.

Proposition 4.1. Let $\mathcal{Z}$ be a finite set, C a set of constraints over the elements in $\mathcal{Z}$, f a sequence function and $\mathcal{A}_{\mathcal{Z},C}$ a consistent algorithm that produces maximal sequences. Denote with $\bar{S} \in 2^{(\mathcal{Z})}$ the first *maximal sequence* found by a GBFS algorithm with evaluation function $f \circ \mathcal{A}_{\mathcal{Z},C}$ on the state transition graph $\mathcal{G}(\mathcal{Z}, C)$ starting from an empty sequence. We have that: $f(\bar{S}) \leq f(\mathcal{A}_{\mathcal{Z},C}(\emptyset))$.[4]

Proof. The proposition follows from the fact that $\mathcal{A}_{\mathcal{Z},C}$ produces a *maximal sequence*, which by definition is a feasible sequence. By contradiction, assume that $f(\bar{S}) > f(\mathcal{A}_{\mathcal{Z},C}(\emptyset))$. This means that all the sequences with a lower f-value expanded before $\bar{S}$ cannot be further extended, contradicting the hypothesis that $\bar{S}$ is the first maximal sequence found. Moreover, because $\mathcal{A}_{\mathcal{Z},C}$ is consistent, GBFS is guaranteed to terminate. □

While this general proposition is simple, we are not aware of any other work in heuristic search planning that uses heuristics based on feasible solutions.

Lemma 4.1. Given the planning task of a S&T problem defined by $\mathcal{C}, \mathcal{O}, \Gamma, D$, the first *maximal sequence* of actions found by GBFS with heuristic function given by Algorithm 3

4. GBFS prioritizes states with lower value of the evaluation function.

has an objective function value that is greater or equal than the objective function of the greedy algorithm (Algorithm 1).

Proof. First, we note that the heuristic value is $1 - P(S)$. We need to show that Algorithm 1 produces maximal sequence and it is consistent. The algorithm produces maximal sequences by definition, as it terminates when no more search patterns can be inserted without violating temporal constraints. In principle, it is not consistent since the observer assignment is not optimal. However, we can force consistency by caching the solutions of sequences explored by GBFS: when GBFS adds a search pattern σ we simply take the minimum between $f(A_{\mathcal{Z},\mathcal{C}}(S))$ and $f(A_{\mathcal{Z},\mathcal{C}}(S \cup \{\sigma\}))$ as heuristic value. $\square$

We show the different behavior of the GBFS algorithm when using the default TRPG heuristic and the solution of the greedy algorithm, using Example 2.1. Figure 8a shows that the TRPG heuristic value at the root is 4: the number of actions necessary to add σ_1 in the final solution (*fly* σ_1*, TIL0, doPattern* $\sigma_1 \vdash$*, doPattern* $\sigma_1 \dashv$). Before adding a new search pattern after σ_1, the search algorithm expands states starting with alternative patterns, whose heuristic value is now higher due to the additional actions that the TRPG inserts to achieve a better value of the metric function. It should be noted that the effects of the action on the metric function are calculated by the external solver and during the heuristic calculation such effects are approximated (Bernardini et al., 2017a). The search continues adding search patterns to σ_1 and only investigating other patterns when it cannot append anything to the initial sequence. In contrast, as seen in Figure 8b, our greedy heuristic identifies that adding σ_5 instead of σ_1 could lead to a state with a better objective value, therefore it allows the search algorithm to achieve the solution in fewer states.

4.3.2 Implementation Details

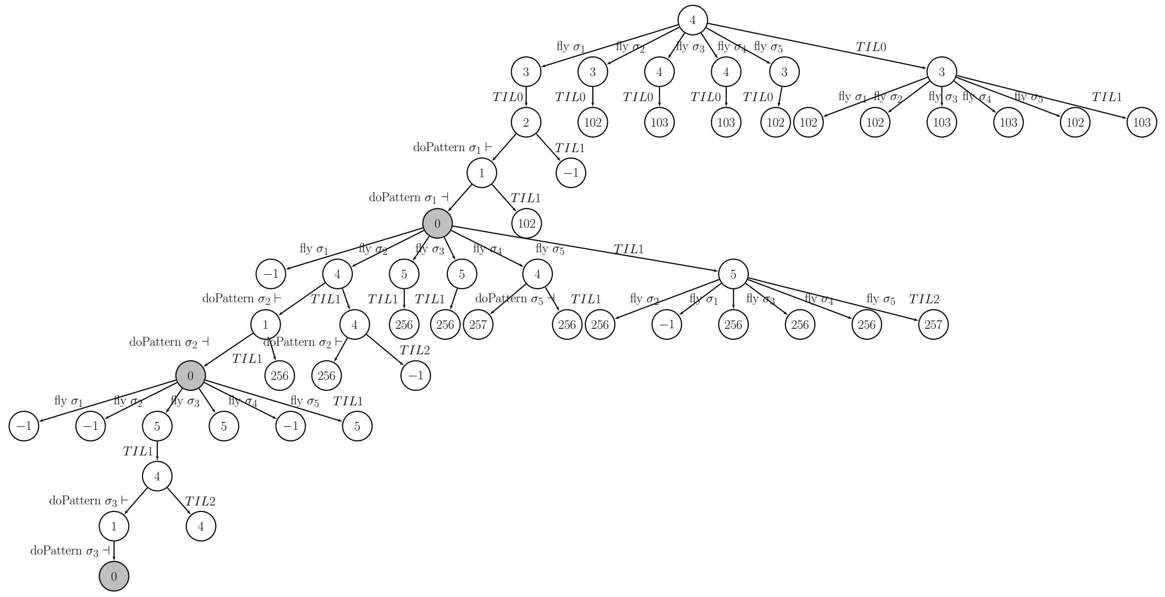
Instead of the greedy algorithm presented in Section 4.1, which is quadratic in the number of search pattern candidates, we consider an accelerated (linear) version of it, following the procedure presented by Minoux (1978) (Algorithm 4). The paper shows that the two greedy algorithms produce the same results when the objective function is submodular. As for the Algorithm 1, we adapt the original algorithm to solve the observer assignment problem and to exclude search patterns that cannot be executed (Algorithm 4: lines 11-17).

5. Experimental Evaluation

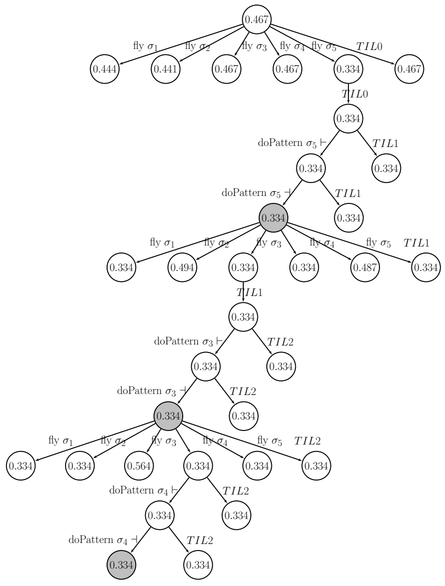
In this section, we present an empirical evaluation of the proposed approaches. We omit the Partially Observable Markov Decision Process (POMDP) approach considered in previous work on single observer because we have empirically shown that it is outperformed by the CP and planning approaches (Bernardini et al., 2017b). We do not expect the POMDP technique to perform better on problems with multiple UAVs as the action space becomes even larger.

First, we consider a set of toy problems and vary the number of search patterns in order to investigate the scaling behavior of different solution approaches. Then, we implement all the approaches in a S&T simulator to evaluate their impact in a realistic scenario.

We run all the experiments on a Xeon 3.5GHz processor machine running Mac OS X Sierra. All solvers are given one minute to generate the plans. Although this



(a) Default heuristic TRPG.



(b) Greedy algorithm-based heuristic. Note that the heuristic value is $1 - P(\Sigma)$.

Figure 8: Example of the two different search space explorations obtained by the default heuristic and our domain specific heuristic. The heuristic value of each state is reported inside the node.

Algorithm 4 Accelerate Greedy Algorithm

```
1: procedure AccelerateUpdateSequence($\widetilde{\mathcal{C}}, \mathcal{O}, \Gamma, D, \Sigma, S$)
2:     $H \leftarrow \emptyset$
3:     for $\sigma \in \widetilde{\mathcal{C}}$ do
4:         $\Delta P(\sigma) = P(\{\sigma\})$
5:     for $i = 0, ..., \max_{\sigma \in \mathcal{C}}(t^+_\sigma + d_\sigma) / \min_{\sigma \in \mathcal{C}}(d_\sigma)|\mathcal{O}|$ do
6:         $H \leftarrow \emptyset$
7:         $\delta \leftarrow 0$
8:         while $H \subset \mathcal{C}$ do
9:             $\sigma \leftarrow \text{argmax}_{\sigma \in \widetilde{\mathcal{C}} \setminus \Sigma} \Delta P(\sigma)$
10:            if $\sigma \in H$ then
11:                $(S, o) \leftarrow$ Assignment($\mathcal{O}, \widetilde{\mathcal{C}}, D, \sigma$)
12:                if $(S^*, o^*) \neq (\emptyset, \emptyset)$ then
13:                    $\sigma^* \leftarrow \sigma$
14:                    $(S^*, o^*) \leftarrow (S, o)$
15:                    break
16:                else
17:                    $\Delta P(\sigma) = 0$
18:            else
19:                $\Delta P(\sigma) = P(\Sigma \cup \{\sigma\}) - P(\Sigma)$
20:                $H \leftarrow H \cup \{\sigma\}$
21:        $\delta \leftarrow \Delta P(\sigma^*)$
22:        if $\delta \leq 0$ then return $\Sigma, S$
23:        else
24:            $\Sigma \leftarrow \Sigma \cup \{\sigma^*\}$
25:            $S^{o^*} \leftarrow S^*$
    return $\Sigma, S$
```

time limitation compromises the ability for the CP solver and the AI planner to find optimal solutions, in surveillance operations, time is a critical factor. It is therefore preferable to find suboptimal solutions in a short period of time, than produce provably optimal solutions. We solve the CP model using IBM ILOG CPLEX CP Optimizer v12.8.0 (Laborie et al., 2018).

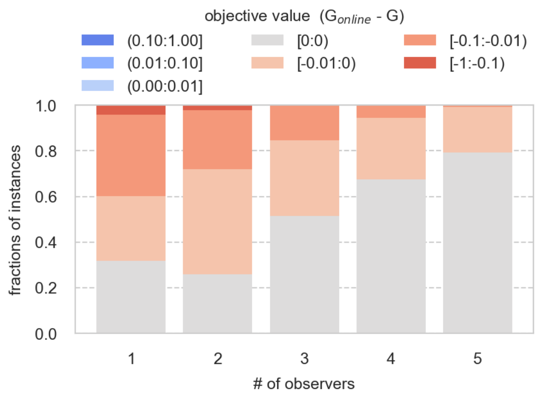
5.1 Scaling Behavior of the Approaches

For the first experiment, we randomly generate 40 instances with 30 candidate search patterns. For each instance, we reduce the number of search patterns to a minimum of six, and we vary the number of observers, from a minimum of one to a maximum of five. For this set of experiments, we record the objective values of the different solution approaches as a function of the number of search patterns.

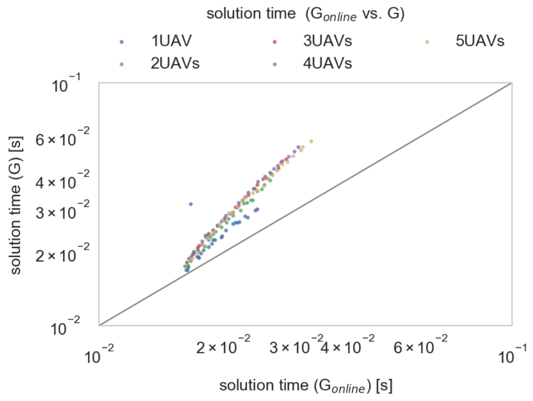
5.1.1 Comparison of the Greedy Solutions

We first compare the two greedy algorithms that we consider. We identify with G the accelerated version (Algorithm 4) and with G_{online} the simplified version with the Right Assignment Algorithm. Figure 9a shows the fraction of problem instances for which the difference in the objective values found by G and G_{online} is within given intervals. Figure 9b reports the execution time of every instance for the two algorithms. As shown in the plots, G finds equal or better quality solutions than G_{online} for every problem instance. Although the execution time of G is higher than G_{online}, the algorithms always run within 0.07 seconds. The difference between the two algorithms in the quality of the solutions decreases as the number of observers increases. The increase in the number of observers makes the problem less constrained, making it more likely to find an available observer for the execution of a search pattern at the end of the current sequence.

As already noted, a greedy algorithm is theoretically guaranteed to produce solutions close to optimality for problems that require the maximization of a submodular function. In our problem, this theoretical guarantee does not hold because of the presence of temporal constraints. We argue that, in practice, these constraints do not affect the performance of the greedy algorithm significantly. To verify our claim, we take our problem instances and modify the starting time-window $[t^-_\sigma, t^+_\sigma]$ associated with each search pattern $\sigma \in \Sigma$ by scaling the length of the interval by a constant value. When the time-windows are narrow, the sequence of search patterns is more constrained, which means that it is more likely that the best pattern chosen by the greedy approach cannot be added to the solution because it does not respect the temporal constraints. With wide time-windows, the best pattern chosen by the algorithm can almost always be added to the solution, making the problem very similar to a simple maximization problem. In Figure 10, we show the average objective value obtained as a function of the scaling factor of the starting time-windows. As shown in the figure, when the problem is highly constrained, the greedy algorithm finds solutions within 10% from the optimal value. As the length of starting time-windows become larger, the greedy algorithm finds increasingly better quality solutions. When considering problems with several observers, the effect of the temporal constraints is even less visible, as at every step multiple vehicles are available to execute the search pattern with the best impact on the objective value.

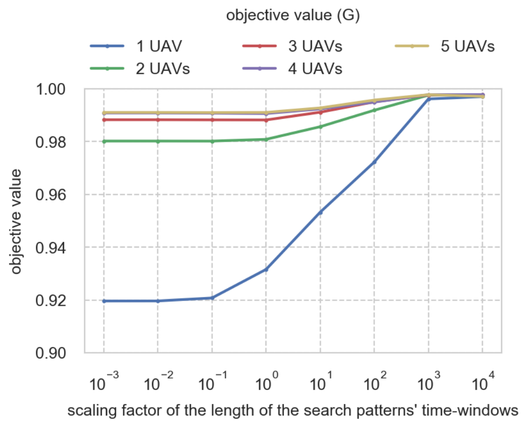


(a) Difference in the objective values of G and G_{online} categorized into buckets. [x:y) = "Fraction of instances where the difference in the objective values obtained by G and G_{online} is within [x:y)".

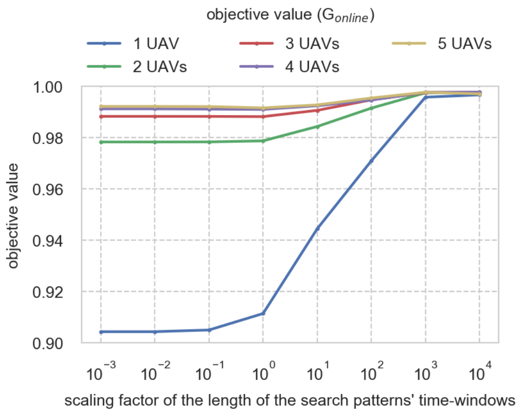


(b) Comparison of the execution time of the accelerated (G) and online (G_{online}) greedy algorithms. The execution time of G is consistently higher than G_{online}, but both algorithms always run within 0.06 s.

Figure 9: Comparison between the accelerated (G) and online (G_{online}) greedy algorithms.



(a) Effect of the temporal constraints on the objective function for G.



(b) Effect of the temporal constraints on the objective function for G_{online}.

Figure 10: Average of the solution quality found by the greedy algorithm for problems with different scaled length of starting time-windows.

5.1.2 Comparison of CP Models

We now compare the two CP models presented in Section 4.2. For the first model (Figure 5), previously proposed by Bernardini et al. (2017b)), we use a time-discretization of 10 and 100 seconds, and we indicate these two variations as $CP\Delta_{10}$ and $CP\Delta_{100}$, respectively. The new CP model in Figure 7 using interval variables is referred to as CP_{iv}.

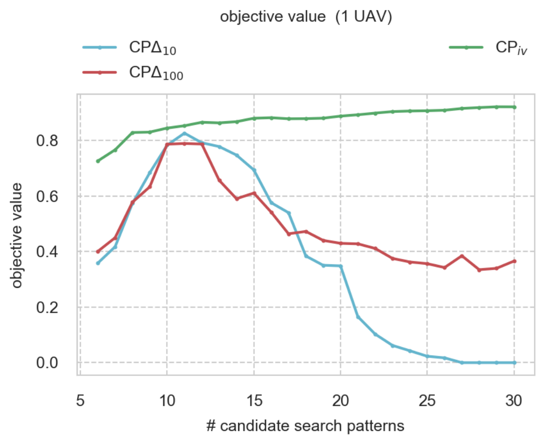


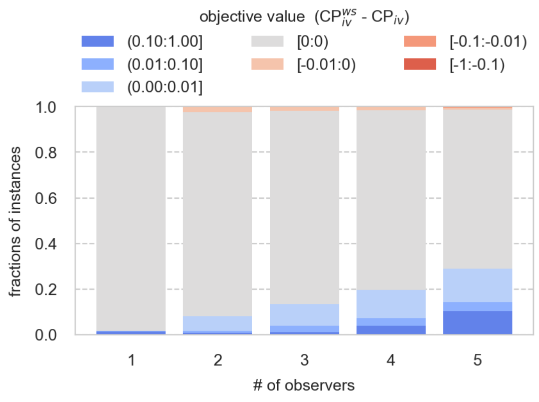
Figure 11: Comparison between different CP models for problems with 1 observer.

Figure 11 shows the average objective value as a function of the number of search patterns for problems with one observer. As expected, $CP\Delta_{10}$ and $CP\Delta_{100}$ produce relatively good quality solutions for problems with a small number of search patterns, but $CP\Delta_{10}$ cannot produce any solution when the number of search patterns increases. CP_{iv} produces better quality solutions than both $CP\Delta_{10}$ and $CP\Delta_{100}$. This improvement can be attributed to two reasons. First, CP_{iv} is more efficient in the limited time allocated to solve the problems because it needs fewer variables and exploits the inference power of the interval variables. In addition, when using $CP\Delta_{10}$ and $CP\Delta_{100}$, some of the search patterns cannot be executed due to the coarse-time discretization and, as a consequence, they do not contribute to the objective value. This phenomenon never happens when we apply CP_{iv}.

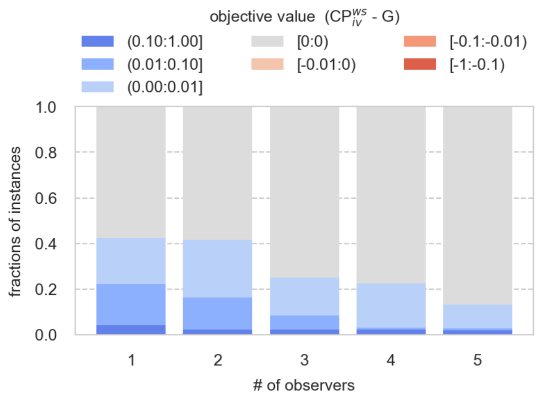
We can observe the effect of the warm start on the CP_{iv} model in Figure 12. We call CP_{iv}^{ws} the model that is warm-started with the G solution. As shown in the figure, the warm start has a little effect on smaller problems (1 observer) since they are simple enough to be optimally solved by the CP model alone. As the problem size increases, the warm-started model can find solutions that are better than CP_{iv} and G, but for larger problems, the warm-started CP model only marginally improves on the G solution.

5.1.3 Comparison of the Planning Approaches

We compare the solutions obtained by the planner with the standard TRPG heuristic (Π^{TRPG}) and our greedy-based heuristic (Π^{G}). In Figure 13a, we compare Π^{TRPG} and Π^{G}, while in Figure 13b we compare Π^{G} and G. As expected, the planner with the greedy-based heuristic outperforms the standard TRPG heuristic and improves on the initial greedy solution. We observe that, for problems with one observer, the difference between Π^{G} and Π^{TRPG} is relatively small and increases as the number of observers increases. When more observers are present, the TRPG heuristic produces particularly poor solutions, with one search pattern for each observer. This is because the heuristic is based on the length of the relaxed plan and the search explores all the states with the same number of search patterns,

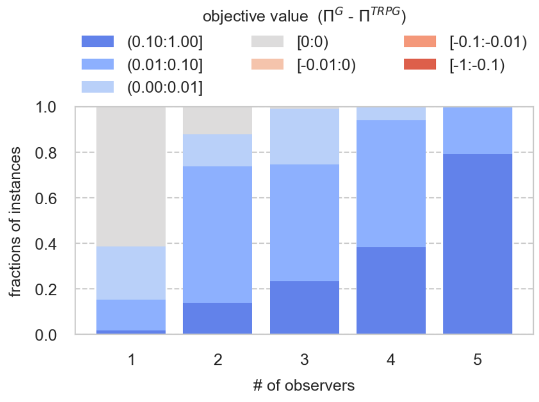


(a) Difference in objective value of CP_{iv} and CP_{iv}^{ws} categorized into buckets.

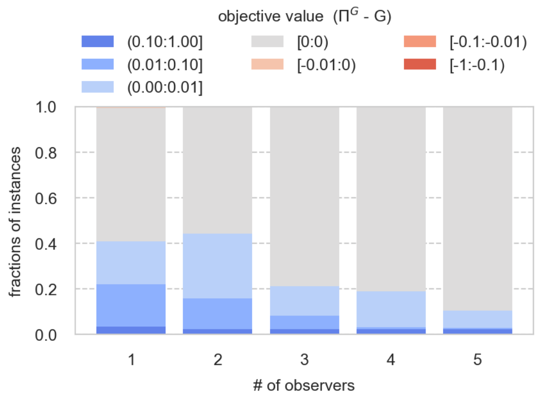


(b) Difference in objective value of G and CP_{iv}^{ws} categorized into buckets.

Figure 12: Comparison between CP_{iv}, CP_{iv}^{ws} and G.



(a) Difference in the objective value of Π^{TRPG} and Π^G categorized into buckets.



(b) Difference in the objective value of G and Π^G categorized into buckets.

Figure 13: Comparison between different planning heuristics.

before adding a new one. As in the CP model, the improvement of Π^G with respect to the greedy solution decreases for larger problems.

5.1.4 Comparison of the Approaches

We now compare the different approaches: greedy, CP and AI planning. In Figure 14 and Table 2, we show a summary of the average solution quality $P(S)$ found by each method considering problems with a number of observers varying from one to five. The approaches are ranked according to the sum of the considered score function for problems with a different number of search patterns.

From the figure, it emerges that the best overall approaches are CP^{ws}_{iv} and Π^G, which have comparable performance, as shown also in Figure 15. Both approaches improve on the greedy solution. We can observe that neither Π^{TRPG} nor any of the CP models scale to large size instances without the help of the greedy solution. From Table 2, it also emerges that, as the number of observers increases, the greedy algorithm performs very similarly to the best approaches, i.e. Π^G and CP^{ws}_{iv}.

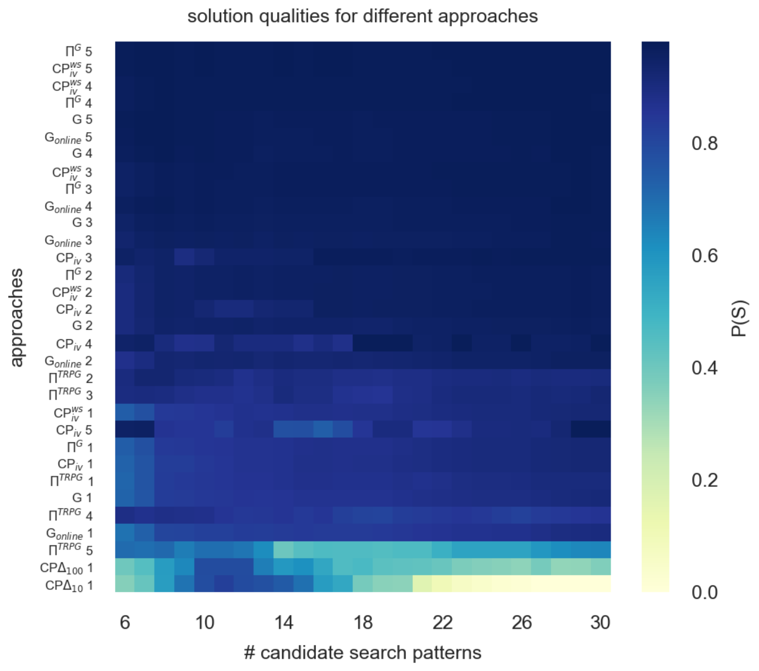


Figure 14: Average solution quality for problems with a different number of search patterns and observers. Each box is an average over 40 instances and the color indicates the measure of the solution quality.

5.2 Simulation

We conducted a set of experiments using a simulator of fixed-wing UAVs operating in a square area of about 100 kilometers width. The target is mobile, moving from a fixed origin to a random destination located in one of the urban centers of Scotland. The target follows a route acquired using GraphHopper (Karich, 2015). When it realizes that it is observed, the target can dynamically change its route to prefer concealed paths (Bernardini et al., 2017b). At the beginning of a mission, one UAV tracks the target, while the others are located at a base station. When the tracking UAV loses the target, it communicates that to the ground station and the search phase is initiated. The ground station performs the MCS to generate the candidate search patterns. The search area considered in the MCS is an angular sector from the LKP. When only one observer is available the angle is 120° and it increases uniformly as more observers are available, up to a maximum of 180° for 5

# SP	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Π^G 5	0.976	0.982	0.981	0.977	0.978	0.977	0.976	0.977	0.978	0.978	0.978	0.979	0.977	0.978	0.977	0.977	0.978	0.979	0.979	0.980	0.981	0.981	0.980	0.981	0.979
CP_{iv}^{ws} 5	0.976	0.982	0.981	0.977	0.978	0.977	0.976	0.977	0.978	0.977	0.978	0.979	0.976	0.978	0.977	0.976	0.975	0.979	0.979	0.980	0.981	0.981	0.981	0.981	0.979
CP_{iv}^{ws} 4	0.969	0.978	0.979	0.974	0.976	0.976	0.975	0.975	0.977	0.977	0.977	0.977	0.975	0.976	0.975	0.976	0.977	0.978	0.978	0.979	0.979	0.980	0.980	0.980	0.978
Π^G 4	0.969	0.977	0.979	0.974	0.976	0.976	0.975	0.975	0.976	0.976	0.977	0.978	0.975	0.976	0.975	0.976	0.977	0.978	0.978	0.979	0.980	0.980	0.980	0.980	0.978
G 5	0.975	0.981	0.981	0.976	0.978	0.974	0.973	0.968	0.971	0.971	0.972	0.972	0.970	0.971	0.973	0.974	0.973	0.973	0.974	0.975	0.976	0.975	0.980	0.981	0.979
G_{online} 5	0.973	0.979	0.980	0.975	0.977	0.973	0.972	0.967	0.970	0.970	0.971	0.971	0.969	0.970	0.972	0.973	0.972	0.973	0.973	0.974	0.975	0.975	0.980	0.980	0.978
G 4	0.968	0.976	0.978	0.974	0.976	0.972	0.971	0.966	0.969	0.969	0.970	0.971	0.968	0.969	0.971	0.973	0.971	0.972	0.973	0.974	0.974	0.974	0.979	0.979	0.978
CP_{iv}^{ws} 3	0.952	0.964	0.972	0.968	0.971	0.971	0.970	0.970	0.973	0.973	0.973	0.973	0.971	0.972	0.971	0.972	0.974	0.976	0.976	0.976	0.978	0.978	0.978	0.978	0.976
Π^G 3	0.954	0.964	0.972	0.968	0.971	0.970	0.969	0.969	0.972	0.972	0.973	0.973	0.971	0.972	0.971	0.972	0.974	0.975	0.976	0.977	0.978	0.978	0.978	0.978	0.976
G_{online} 4	0.963	0.972	0.975	0.970	0.973	0.970	0.969	0.963	0.967	0.967	0.968	0.968	0.965	0.966	0.968	0.969	0.969	0.971	0.971	0.972	0.973	0.973	0.979	0.979	0.977
G 3	0.950	0.963	0.970	0.965	0.969	0.965	0.965	0.959	0.963	0.964	0.965	0.965	0.963	0.964	0.967	0.968	0.967	0.969	0.969	0.970	0.971	0.971	0.977	0.977	0.975
G_{online} 3	0.936	0.953	0.962	0.957	0.962	0.959	0.960	0.952	0.959	0.959	0.960	0.959	0.957	0.959	0.961	0.962	0.962	0.966	0.966	0.967	0.968	0.968	0.974	0.975	0.973
CP_{iv} 3	0.952	0.940	0.947	0.896	0.923	0.946	0.946	0.945	0.948	0.948	0.973	0.973	0.970	0.971	0.970	0.971	0.974	0.973	0.974	0.971	0.976	0.976	0.975	0.977	0.975
Π^G 2	0.910	0.934	0.950	0.947	0.952	0.955	0.955	0.953	0.957	0.958	0.959	0.959	0.957	0.958	0.959	0.962	0.964	0.966	0.967	0.967	0.969	0.969	0.970	0.971	0.968
CP_{iv}^{ws} 2	0.905	0.931	0.949	0.946	0.951	0.954	0.955	0.952	0.957	0.959	0.959	0.960	0.958	0.959	0.960	0.962	0.964	0.966	0.966	0.968	0.969	0.969	0.970	0.971	0.968
CP_{iv} 2	0.904	0.931	0.949	0.946	0.927	0.906	0.907	0.928	0.933	0.934	0.959	0.960	0.958	0.959	0.960	0.962	0.963	0.966	0.967	0.967	0.969	0.969	0.968	0.970	0.968
G 2	0.902	0.927	0.946	0.941	0.947	0.947	0.949	0.940	0.945	0.946	0.945	0.946	0.943	0.943	0.947	0.953	0.954	0.958	0.958	0.959	0.960	0.960	0.966	0.967	0.965
CP_{iv} 4	0.945	0.953	0.908	0.876	0.879	0.927	0.902	0.902	0.903	0.878	0.903	0.879	0.974	0.975	0.975	0.950	0.952	0.976	0.952	0.954	0.976	0.954	0.953	0.955	0.976
G_{online} 2	0.874	0.898	0.929	0.925	0.934	0.931	0.933	0.922	0.931	0.928	0.927	0.930	0.927	0.930	0.934	0.939	0.943	0.948	0.948	0.949	0.949	0.953	0.960	0.961	0.959
Π^{TRPG} 2	0.902	0.929	0.930	0.911	0.905	0.898	0.873	0.887	0.902	0.895	0.894	0.886	0.883	0.879	0.888	0.886	0.899	0.903	0.903	0.905	0.898	0.907	0.902	0.904	0.906
Π^{TRPG} 3	0.899	0.903	0.901	0.886	0.875	0.874	0.868	0.882	0.907	0.893	0.890	0.867	0.860	0.858	0.883	0.888	0.901	0.909	0.909	0.912	0.907	0.916	0.911	0.916	0.925
CP_{iv}^{ws} 1	0.737	0.776	0.842	0.845	0.853	0.861	0.871	0.868	0.873	0.881	0.882	0.878	0.879	0.880	0.888	0.892	0.898	0.904	0.906	0.907	0.909	0.915	0.918	0.921	0.921
CP_{iv} 5	0.952	0.957	0.861	0.855	0.854	0.832	0.874	0.879	0.778	0.782	0.734	0.784	0.854	0.903	0.903	0.855	0.856	0.880	0.905	0.905	0.906	0.904	0.927	0.977	0.975
Π^G 1	0.739	0.776	0.842	0.845	0.853	0.857	0.866	0.864	0.869	0.876	0.878	0.875	0.875	0.877	0.884	0.892	0.898	0.904	0.906	0.907	0.909	0.915	0.918	0.921	0.921
CP_{iv} 1	0.726	0.766	0.829	0.830	0.845	0.853	0.865	0.863	0.868	0.880	0.882	0.878	0.879	0.880	0.888	0.892	0.898	0.904	0.906	0.907	0.909	0.915	0.918	0.921	0.921
Π^{TRPG} 1	0.718	0.764	0.834	0.842	0.849	0.852	0.861	0.859	0.867	0.875	0.876	0.873	0.873	0.874	0.882	0.874	0.880	0.892	0.894	0.896	0.898	0.906	0.907	0.908	0.906
G 1	0.722	0.761	0.833	0.838	0.844	0.848	0.857	0.851	0.858	0.865	0.863	0.864	0.864	0.864	0.872	0.880	0.882	0.889	0.890	0.891	0.892	0.899	0.909	0.913	0.913
Π^{TRPG} 4	0.886	0.873	0.884	0.880	0.877	0.859	0.847	0.850	0.855	0.843	0.854	0.826	0.815	0.810	0.830	0.838	0.845	0.852	0.845	0.828	0.818	0.836	0.847	0.855	0.862
G_{online} 1	0.689	0.730	0.806	0.812	0.822	0.824	0.835	0.830	0.838	0.841	0.838	0.838	0.839	0.840	0.848	0.856	0.860	0.867	0.869	0.872	0.873	0.886	0.895	0.899	0.903
Π^{TRPG} 5	0.705	0.697	0.712	0.663	0.697	0.696	0.682	0.623	0.405	0.445	0.468	0.460	0.463	0.456	0.464	0.464	0.513	0.554	0.554	0.554	0.556	0.594	0.619	0.634	0.638
$\text{CP}\Delta_{100}$ 1	0.400	0.450	0.578	0.633	0.786	0.789	0.787	0.657	0.590	0.611	0.542	0.464	0.472	0.440	0.430	0.428	0.411	0.375	0.363	0.356	0.342	0.385	0.335	0.340	0.366
$\text{CP}\Delta_{10}$ 1	0.358	0.417	0.574	0.684	0.783	0.826	0.792	0.778	0.747	0.694	0.576	0.540	0.384	0.351	0.348	0.166	0.103	0.062	0.043	0.023	0.017	0.000	0.000	0.000	0.000

Table 2: Average solution quality for problems with a different number of search patterns and observers. Each box is an average over 40 instances.

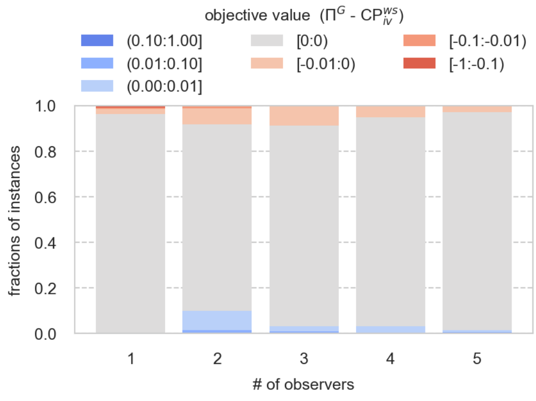


Figure 15: Difference in the objective value of Π^G and CP_{iv}^{ws} categorized into buckets.

observers. The MCS considers 20 time points and, for each time point, it generates up to $|\mathcal{O}| + 1$ search pattern candidates. Once the candidate search patterns are generated, the ground station calls a solver (greedy, AI planning, or CP) to find the sequence of search patterns to execute and the plan is dispatched to each UAV, which executes the search patterns assigned to it. The first UAV that finds the target takes control of the tracking operations, while the other UAVs abandon the search until the target is lost again and a new search phase is activated or the mission is concluded. The mission is declared successful if the target is tracked at the moment of its arrival to the destination. For every configuration, we simulate 500 missions.

As a baseline, we use a simple random policy, where, starting from the LKP and knowing the estimated direction of the target, each observer performs a random walk. As in the

MCS, we consider 20 time points. At each time point, the observer moves from its previous position in a random direction and distance. The direction and the distance are chosen using a uniform distribution between two fixed angles and the two minimum and maximum distances estimated by the minimum and maximum velocity of the target.

As an alternative approach, we consider a static policy defined by our initial industrial partner (Bernardini et al., 2013). The policy for one observer consists in performing a spiral search around the LKP of the target and then executing a lawnmower pattern that has a longitudinal axis aligned with the estimated direction of the target. We extended this strategy to multiple observers by adding one lawnmower pattern for each additional UAV. The lawnmower is obtained by rotating the longitudinal axes of the first added lawnmower by a fixed angle.

Table 3 reports the results of the simulation. For each method, we show the average success rate over all the simulated missions. The final success rate is given by the execution of multiple plans. In order to show the correlation between the objective function found by each solver and the success rate, we report the success rate of the first executed plan. As expected, solvers that produce plans with higher objective values have a higher success rate. Finally, we report the average tracking time of the missions and the standard error. The relatively high values of the errors reflect the high variability of each run of the simulation caused by the many choices that contribute to the target behavior (i.e., the target chooses its destination, the path to follow, the evasion level of its behavior, velocity, etc.).

As shown in the table, when only one observer is present, both Π^{TRPG} and CP_{iv} and their enhancements Π^G, CP_{iv}^{ws}, obtain the best performance, succeeding in 1/3 of the missions. The greedy approach performs almost as well as planning and CP, achieving a success rate of 30%. The static and the online policies follow with a similar success rate (25.2% and 23.8%, respectively), while the random approach gives the worst results, successfully tracking the target until its destination in 17.0% of the cases. When we have two observers, all techniques improve their success rate and a similar behaviour as before can be observed: Π^G and CP_{iv}^{ws} obtain the highest success rate, followed by G. While the success rates of Π^{TRPG} and CP_{iv} do not increase as much as the other techniques, but they both still perform better than the static, online and random policies. As we increase the number of observers up to five, G becomes similar to Π^G and CP_{iv}^{ws} and dominates both Π^{TRPG} and CP_{iv}, as well as the online, static and random policies. The static policy becomes competitive with our optimization techniques only once enough observers are deployed.

6. Discussion

As noted in Section 1 the primary contributions of this paper are application-oriented: the proof of submodularity allowing the exploitation of the greedy algorithm, the development of a novel CP model and an extension of the existing AI planning approach for the problem, the integration of the greedy heuristic into the CP and AI planning approaches, and the extensive experimental evaluation. The paper provides a detailed treatment of the combinatorial optimization approach to the search and tracking problem.

Nonetheless, the paper also provides insight beyond the application in at least two areas: the comparison of scheduling and planning approaches for problems with characteristics of

Average % of success rate					
# UAVs	1	2	3	4	5
random	17.0 ± 1.7	24.2 ± 1.9	32.8 ± 2.1	30.0 ± 2.1	34.6 ± 2.1
static	25.2 ± 1.9	30.6 ± 2.1	39.8 ± 2.2	46.0 ± 2.2	52.8 ± 2.2
G_{online}	23.8 ± 1.9	37.8 ± 2.2	42.8 ± 2.2	46.0 ± 2.2	49.4 ± 2.2
G	30.0 ± 2.1	41.8 ± 2.2	$\mathbf{50.8 \pm 2.2}$	50.2 ± 2.2	$\mathbf{53.6 \pm 2.2}$
CP_{iv}	32.8 ± 2.1	41.8 ± 2.2	40.6 ± 2.2	30.4 ± 2.1	33.4 ± 2.1
CP_{iv}^{ws}	33.0 ± 2.1	44.8 ± 2.2	47.2 ± 2.2	$\mathbf{51.8 \pm 2.2}$	52.0 ± 2.2
Π^{TRPG}	$\mathbf{34.6 \pm 2.1}$	38.0 ± 2.2	26.6 ± 2.0	11.0 ± 1.4	7.2 ± 1.2
Π^{G}	34.2 ± 2.1	$\mathbf{45.6 \pm 2.2}$	49.6 ± 2.2	49.0 ± 2.2	48.6 ± 2.2
Average % of success rate after the execution of the first plan					
# UAVs	1	2	3	4	5
random	45.5 ± 2.3	53.8 ± 2.3	68.9 ± 2.1	70.1 ± 2.1	72.8 ± 2.0
static	60.7 ± 2.2	70.1 ± 2.1	76.9 ± 1.9	84.3 ± 1.6	89.8 ± 1.4
G_{online}	68.8 ± 2.1	82.7 ± 1.7	89.6 ± 1.4	93.5 ± 1.1	93.9 ± 1.1
G	75.8 ± 1.9	87.9 ± 1.5	92.2 ± 1.2	94.5 ± 1.0	$\mathbf{96.5 \pm 0.8}$
CP_{iv}	78.9 ± 1.9	84.2 ± 1.7	84.0 ± 1.7	73.8 ± 2.0	74.4 ± 2.0
CP_{iv}^{ws}	79.1 ± 1.8	$\mathbf{88.6 \pm 1.5}$	92.0 ± 1.2	$\mathbf{94.9 \pm 1.0}$	$\mathbf{96.5 \pm 0.8}$
Π^{TRPG}	77.9 ± 1.9	83.4 ± 1.7	64.4 ± 2.2	27.0 ± 2.0	20.4 ± 1.8
Π^{G}	$\mathbf{79.8 \pm 1.8}$	88.1 ± 1.5	$\mathbf{92.4 \pm 1.2}$	90.6 ± 1.3	90.6 ± 1.3
Average number of losses					
# UAVs	1	2	3	4	5
random	1.52 ± 0.04	1.77 ± 0.05	2.10 ± 0.06	2.20 ± 0.06	2.28 ± 0.06
static	1.73 ± 0.04	1.97 ± 0.05	2.43 ± 0.06	2.66 ± 0.06	2.80 ± 0.06
G_{online}	1.85 ± 0.04	2.14 ± 0.05	2.64 ± 0.06	2.79 ± 0.06	2.80 ± 0.06
G	2.01 ± 0.05	2.37 ± 0.06	2.78 ± 0.06	2.91 ± 0.06	2.99 ± 0.06
CP_{iv}	2.04 ± 0.05	2.29 ± 0.06	2.61 ± 0.06	2.24 ± 0.06	2.28 ± 0.06
CP_{iv}^{ws}	2.05 ± 0.05	2.42 ± 0.06	2.80 ± 0.06	$\mathbf{2.93 \pm 0.06}$	$\mathbf{3.00 \pm 0.06}$
Π^{TRPG}	1.99 ± 0.05	2.22 ± 0.05	2.05 ± 0.05	1.33 ± 0.03	1.23 ± 0.03
Π^{G}	$\mathbf{2.07 \pm 0.05}$	$\mathbf{2.39 \pm 0.06}$	$\mathbf{2.82 \pm 0.06}$	2.83 ± 0.06	2.84 ± 0.06
Average tracking time is seconds					
# UAVs	1	2	3	4	5
random	1405 ± 40	1538 ± 42	1701 ± 44	1721 ± 45	1745 ± 46
static	1599 ± 37	1699 ± 38	1876 ± 45	2065 ± 43	2163 ± 43
G_{online}	1627 ± 37	1836 ± 37	2037 ± 38	2149 ± 35	2126 ± 37
G	1773 ± 38	1996 ± 36	$\mathbf{2203 \pm 37}$	2266 ± 33	$\mathbf{2241 \pm 35}$
CP_{iv}	1774 ± 36	1928 ± 38	2015 ± 40	1754 ± 45	1728 ± 45
CP_{iv}^{ws}	1775 ± 36	$\mathbf{2039 \pm 37}$	2194 ± 37	$\mathbf{2273 \pm 33}$	2236 ± 35
Π^{TRPG}	1790 ± 38	1861 ± 36	1607 ± 45	1045 ± 41	868 ± 35
Π^{G}	$\mathbf{1797 \pm 36}$	2026 ± 35	2180 ± 36	2170 ± 37	2115 ± 39

Table 3: Average % of success rate, average % of success rate after the execution of the first plan, and the average tracking time using different solvers. For each parameter we report the average over 500 missions and the standard error.

both and the tension between domain-dependent planning and domain-independent planning.

Planning vs. Scheduling. The fundamental difference between planning and scheduling is the existence of a fixed number of operations in the latter.[5] This difference is not only reflected in the different complexity classes of the problems but also in the fact that most of the inference done in CP scheduling techniques is only sound because of the inability to dynamically add new operations.

For problems with a non-fixed number of operations that require detailed numeric, resources and temporal reasoning, we are left with the challenge of the richer model of AI planning, with relatively weak techniques for such reasoning tasks versus the creation of a scheduling model with a valid upper bound on the number of optional operations. This challenge is what we have investigated here and in previous work on diverse application problems (e.g., Tran et al. (2017), Booth, Do, Beck, Rieffel, Venturelli, and Frank (2018)). It seems clear that as the problem size scales, planning will be the only viable option: the CP models become too large. However, we have demonstrated here and elsewhere (e.g., Tran et al. (2017)), that, if it is possible to fix the maximum number of actions that are considered, realistic-sized problems are within the reach of CP.

Another challenging aspect of the problem tackled in this paper is the representation of the objective function. From a CP perspective, the recursive structure of the objective function results in a weak propagation of its upper and lower bounds, compromising the performance of the solver. In planning, the challenge is overcome by augmenting the solver with domain-specific components that update the value of the objective function and guide the search using a customized heuristic.

The relationship between planning and optimization and, more fundamentally, state-based and constraint-based knowledge representations, is a key area for research into problem solving. Some valuable and perhaps foundational work has already been done (Pommerening, 2017) but much is left to do.

Planning vs. Heuristic Search. Our planning model is posed in PDDL and is solved by a domain-independent planner. However, the primary reason for the strong performance of the model is the use of the domain-specific heuristic function and an external solver for the nonlinear objective function. A narrow interpretation of AI planning as domain-independent planning may therefore result in the conclusion that our "planning" approach is more accurately described as heuristic search.[6]

However, we would like to argue that the contributions of our work and the value of this style of planning/heuristic search (PHS) research does not depend on whether the work is planning or heuristic search. Our goal, beyond developing the search-and-tracking application, is to extend the power and use of PHS technology. One way to do this is through comparing the performance of PHS to other state-of-the-art techniques on specific problems.[7] A fair comparison with other technology requires the same level of domain-dependent specializations – otherwise the more general purpose technique is likely to always perform worse and not because of the underlying problem solving approach.

5. In scheduling such elements are variously referred to as jobs, activities, tasks, or operations, while in planning they are typically called (ground) actions.
6. Indeed, some of the authors are of this opinion.
7. We are, of course, not the only researchers who work in this direction (e.g. Sewell & Jacobson, 2012; Tierney et al., 2012; Kelareva et al., 2013; Tran et al., 2017; Haslum et al., 2018; Booth et al., 2018).

Further, we believe that such an approach is not contrary to the long-term goal of AI planning to develop a general problem solver. First, such comparisons, if done fairly, will likely demonstrate the value of planning technology and increase its use amongst researchers and practitioners who are more focused on solving their problems. In the worst case, if no such value can be demonstrated, the planning community is left with a worthy challenge to understand why and how to improve the technology. Second, the approaches that are developed to achieve strong planning performance in an application can then be generalized and synthesized by subsequent researchers to contribute more directly to domain-independent planning. As a specific example, consider the nonlinear objective function in our search-and-tracking problem. We seek to maximize the probability of finding the target and an increase in the plan length by performing an additional search pattern cannot decrease the probability. Yet most of the existing domain-independent heuristic functions of which we are aware seek to minimize plan length, cost, or makespan. A general problem solver will need to be able to deal with a much richer set of objective functions and, while some work exists on maximization functions via heuristic search (Stern, Kiesel, Puzis, Felner, & Ruml, 2014) and planning (e.g., in planning with preferences Gerevini, Haslum, Long, Saetti, and Dimopoulos (2009), Baier, Bacchus, and McIlraith (2009), Benton, Coles, and Coles (2012) and in over-subscription planning Domshlak and Mirkis (2015)), only a few works exploit submodularity (Chen, Chen, & Weinberger, 2015; Sakaue & Ishihata, 2018). Ideally, domain-dependent research on useful applications will inspire subsequent domain-independent research on common problem characteristics.

7. Related Work

Optimal search is a very complex problem. It is NP-hard even in the simple formulation of finding a stationary target within a grid of cells (Trummel & Weisinger, 1986). Given the inherent complexity of the problem, the literature on search is vast and spread across many disciplines including operations research, graph theory, robotics, control theory and artificial intelligence (AI). It is difficult to compare the methods proposed since the formulations of the problem vary considerably from one work to another. For example, models of target motion range from stationary targets to targets moving according to predictable forces (e.g., wind or currents), to intentional targets, having a goal to achieve or trying to escape. Observers are assumed to have a wide range different sensing, motion and endurance capabilities. There may be one or many targets and observers. The environment can be known in advance or unknown and vary in size and features (e.g., indoors and outdoors). Depending on the search region representation, the sensor types and the possible actions, search can be formulated in discrete or continuous settings. In addition to all these possibilities, search is often combined with other problems. For example, when the target is static and the environment is known, the problem is framed as “search-and-coverage”, while if the environment is unknown, it is “search-and-exploration”. If the target needs to be followed to destination, as in our case, then the problem is “search-and-tracking”. It is not our ambition to summarize all the work on search. Readers can refer to Hollinger (2010) for an interesting survey on search and to Chmaj and Selvaraj (2015) for a survey on the applications of cooperative teams of UAVs, including search-and-tracking. In what follows, we focus on the main approaches to coordinated target search that have inspired our approach and are related to it. We organize these techniques based on their main features, but there are several overlaps among the categories.

7.1 Classical Approaches

Search for lost targets was posed as a research problem in the context of US Navy operations during World War II (Koopman, 1946) and involved providing efficient methods for detecting submarines. Theory of search methods were then used by the US Navy in practical missions to search for objects such as the H-bomb lost in the ocean near Palomares, Spain in 1966 and the submarine Scorpion lost in 1968 (Richardson & Stone, 1971). A few years later, theory of search emerged as a branch of operations research focusing on analytic, optimal solutions for stationary target search (Stone, 1975). Benkoski, Monticino, and Weisinger (1991) offer an annotated bibliography of the search theory literature.

Based on research on search in the maritime domain, the systems CASP (Computer Assisted Search Planning) (Richardson & Discenza, 1980) and its successor SAROPS (Search and Rescue Optimal Planning System) (Kratzke, Stone, & Frost, 2010) have been developed and used by the US Coast Guard since 1974 for search-and-rescue operations involving objects lost at sea. SAROPS has two main components: the simulator and the planner. The simulator produces a time-dependent probability distribution (PD) for the target location using Monte Carlo (MC) particle filtering. The system uses information about the object in distress, provided by human operators, and environmental data, provided by the Environmental Data Server, which gathers environmental data from government and private sources concerning winds, currents, cloud cover, drifters, weather and visibility conditions. Based on the target PD and a collection of available search and rescue units (SRUs), the planner assigns one lawnmower pattern to each SRU. Each SRU executes only one pattern and there is no routing of a vehicle from one pattern to another. The planner seeks to maximize the probability of discovering targets by placing the patterns intelligently: using an iterative strategy, it tries different combinations of locations until a pre-determined period of time has elapsed and, then, reports the best solution. If no SRU finds the target, the simulator generates a new PD by incorporating information about motion (drift) and about the previous unsuccessful search and then the planner generates a new set of patterns. A human always coordinates the two systems and supervises the entire process.

The SAROPS system has some similarities with our approach. For example, it uses MC particle filtering to develop a prior PD for the target's location and it uses lawnmower patterns. However, in contrast to our approach, SAROPS regenerates the target PD at each step and works on a one-step lookahead horizon. As a consequence, it does not incorporate any long term strategic reasoning: based on the target PD and a set of available vehicles, SAROPS assigns one lawnmower to each vehicle, which immediately proceeds to execute it. SAROPS performs geometric reasoning to position the lawnmowers in such a way to maximize the probability of discovering targets. Differently from our approach, the iterative process is not automated, but it involves the presence of a human who supervises the entire process and incorporates simulation and pattern positioning.

7.2 Probabilistic Approaches

A rich area of research in target search is based on a Bayesian construction of the problem where the probability of detection of the targets is used as the objective function for optimal search trajectory generation. This probabilistic approach relies on the use of Recursive Bayesian Estimation (RBE) techniques that recursively update and predict the PD of the target location over time, under the assumption that the prior distribution and the probabilistic motion model of the target are known (Bourgault et al., 2006). Although Bourgault, Furukawa, and Durrant-Whyte (2004) discuss a number of possible constraints that can impact the target motion model (obstacles, force fields and terrain), the target is usually assumed to be subjected to external disturbances and not to move on the basis of its own intentions.

RBE techniques work in two stages: update and prediction. The update stage calculates the posterior distribution of the current state given a prior estimation of the state and a new observation. The prediction stage calculates the PD of the next state using the posterior distribution and the target's motion model. Since the implementation of these two stages is computationally expensive, several approaches have been explored to compute them efficiently, including grid-based methods (Bourgault et al., 2006), particle filters (Chung & Furukawa, 2006), element-based techniques (Furukawa, Durrant-Whyte, & Lavis, 2007) and hybrid particle-element approaches (Lavis & Furukawa, 2008).

The search control problem is solved in a greedy fashion over a very short planning horizon (typically, a one-step lookahead). This myopic planning approach is used to control the computational cost of the technique, which quickly becomes intractable as the number of lookahead steps, the size of the search area, or the number of dimensions of the search space, increases.

The Bayesian filtering approach often tackles S&T at the same time (Furukawa, Bourgault, Lavis, & Durrant-Whyte, 2006) and a unified objective function is used, allowing a vehicle to switch from one mode to the other while maintaining information gained during the previous phases. Other approaches that tackle search-and-tracking as a unified problem can be found in the literature (Tian, Bar-Shalom, & Pattipati, 2008; Hoffmann & Tomlin, 2010; Ryan & Hedrick, 2010; Pitre, Li, & Delbalzo, 2012).

Probabilistic-based S&T has proven successful for problems involving stationary targets or targets moving in small geographical areas, simple motion models, static search spaces and short-term missions (e.g. Furukawa, Mak, Durrant-Whyte, & Madhavan, 2012). However, when these assumptions are not satisfied as in our scenario, RBE techniques are likely to perform poorly due to the high computational cost of accurately maintaining a large state space that includes all the possible positions of the moving targets.

Some probabilistic approaches overlap with information-theoretic methods that formulate search in terms of the estimation of quantities such as Shannon entropy, conditional entropy and Kullback-Leibler divergence. Charrow, Kumar, and Michael (2014), for example, present a particle-filter based method to search for a single moving target using a team of robots equipped with range-only sensors. The robots select actions that maximize the mutual information between the estimate of the target and the future measurements of the robots. Bertuccelli and How (2006) also use particle filtering to approach the problem of search for moving targets when the target motion is poorly known, while Roy and Earnest (2006) use it for search and exploration.

Another probabilistic approach to search is based on formulations using Partially Observable Markov Decision Processes (POMDPs), both for single targets (Hollinger, Kehagias, & Singh, 2007; Hsu, Sun, & Rong, 2008) and multiple targets (Bertuccelli & How, 2006). Carvalho, Teichteil-Konigsbuch, and Lesire (2013) use POMDPs for on-line multi-target detection and recognition missions by an autonomous UAV. Hsu et al. (2008) unify tracking and searching in a unique POMDP model, which is approximatively solved using a sampling technique. He et al. (2010) present an online, forward search, planning-under-uncertainty algorithm for the road-constrained target-tracking problem. In this work, the agent's belief of each target's position is represented as a multi-modal Gaussian belief which is exploited to compute the distribution of posterior beliefs after actions are taken. This analytic computation allows the planner to search deeper by considering policies composed of multi-step action sequences. Deeper searches are beneficial as they result in keeping the targets well-localized for longer periods. This technique has proven successful for small geographical areas, but has not been tested yet on larger regions.

7.3 Combinatorial Optimization

Combinatorial optimization techniques have been explored for the coordinated search domain, although in a more limited way than other techniques. Bernardini et al. (2017, 2016, 2017b) solve S&T problems with a single observer using AI planning and CP. We refer the readers to Sections 4.2 and 4.3 for the main differences between this work and our current approach.

Lau, Huang, and Dissanayake (2006) present a dynamic programming technique to efficiently find a target and a branch-and-bound technique for finding multiple targets (Lau, Huang, & Dissanayake, 2005). These methods suffer from poor scalability when large instances of the problem are considered.

Vidal et al. (2002) consider a pursuit-evasion game with a team of both ground and aerial pursuers and ground evaders. They cast the problem in a probabilistic game theoretic framework and consider two computationally feasible greedy pursuit policies. They do not show performance guarantees on solution quality relative to the optimal one, but demonstrate empirically that their method is efficient in locating targets attempting to evade capture. However, the evaders have only limited escape capabilities and the solution is sensitive to the accuracy of the target motion model.

CP and Mixed Integer Programming have been used to define multiple non-overlapping feasible areas over which UAVs can fly to maximize the total probability of finding stationary targets (Abi-Zeid et al., 2011; Morin et al., 2017).

In the context of AI research and video-game development, Sun, Uras, Koenig, and Yeoh (2012) propose the first incremental anytime search algorithm for moving-target search in known terrain. Their technique is very effective in small grid environments, but its scalability has not yet been studied.

7.4 Path Planning

The simplest path-planning strategies for search focus on visiting known locations while minimizing distance traveled or the time to finish the mission (Bellingham, Tillerson, Alighanbari, & How, 2002; Enright, Savla, Frazzoli, & Bullo, 2009). More sophisticated methods consider searching for one or multiple targets in unknown positions and aim to maximize the probability of detection. In these works, the target is no longer considered after it is found. Chung and Carpin (2011), for example, present a method to search for and localize a stationary target using a team of flying robots. The estimation is done using a quad-tree decomposition of the environment to maximize computational efficiency over large search areas. The objective is to maximize the probability of detection divided by the control effort.

Considerable work has been devoted to devising efficient path-planning methods for UAVs involved in search-and-rescue operations. Lin and Goodrich (2014), for example, consider the problem of wilderness search and rescue where mini-UAVs are used to locate missing persons. They propose a new family of path-planning algorithms that use a special spatial representation, the task difficulty map, to model the sensor detection probability. The map is used during planning with a new heuristic, the mode goodness ratio, to prioritize search sub-areas that present a higher probability of rediscovering the target. This work has a number of similarities with our approach. Both techniques aim to produce efficient flight maneuvers for the UAV to maximize the probability of finding the target in the face of sensor limitations and environmental constraints. We exploit probabilistic reasoning to guide the planner to visit the most promising sub-regions first. However, our approach is not devised specifically for path-planning, but rather tackles the entire decision-making process that underpins the behavior of the UAVs. As well as trajectories, when necessary, we can plan additional actions for the UAVs, such as performing search patterns, and potentially refilling depleted resources and avoiding localization failures. In addition, we consider moving targets in wide spaces (hundreds of square kilometers), while Lin and Goodrich (2014) assume the target to be stationary within a limited space.

8. Conclusions

In this paper, we consider the problem of search-and-tracking for a single, mobile target with multiple UAVs in a large geographical region and for an extended period of time. S&T is a difficult problem that is often encountered in real-world missions such as disaster response, surveillance, law enforcement, and monitoring. So far, progress in the area of S&T has been limited to unrealistic scenarios or missions with a single UAV. In our work, we take up the challenge of developing techniques that can scale to solve real-world problems.

We cast S&T as a deterministic combinatorial problem, departing from the traditional formulation as a probabilistic continuous problem, and we prove that the objective function of our formulation is submodular. Starting from this observation, we develop three different optimization approaches to tackle S&T. The first is a greedy algorithm, which works very well in our case even if the temporal constraints in our problem do not admit the sub-optimality guarantee that holds for algorithms that maximize pure submodular functions. We also apply AI planning and constraint programming (CP) to S&T. We extend previous work in the area to multiple observers, improve existing models and use the solutions provided by the greedy algorithm for warm starting the CP model and for a new domain-dependent heuristic for the planner popf-tif.

Our experimental evaluation shows that the greedy solution is fast and effective, especially when there are multiple observers available for search. However, when resources are limited, using more powerful optimization techniques pays off. Since our problems are large and complex, CP and AI planning, however, are not efficient enough if they are not enhanced by the greedy solution. It is in the integration of the greedy algorithm with CP or AI planning that a winning strategy can be found.

In our future work, we will consider missions where multiple targets need to be discovered and tracked to their destinations. We expect that the introduction of multiple targets will partially erode the efficiency of our solution approaches and thus, to control the complexity of the problem, we plan to explore powerful decomposition techniques such as logic-based Benders decomposition (Hooker & Ottosson, 2003) and branch-and-check (Beck, 2010). We also plan to study the similarities between our scenario and the combined location-routing problem from the operations research (OR) literature (Min, Jayaraman, & Srivastava, 1998; Fazel-Zarandi, Berman, & Beck, 2013).

Acknowledgments

We would like to thank the anonymous reviewers whose valuable feedback helped improve the final paper. This research has been supported by the Engineering and Physical Sciences Research Council (grant EP/S0164473/1), Natural Sciences and Engineering Research Council of Canada, Royal Holloway University of London and Massachusetts Institute of Technology.

Appendix A. Generation of Candidate Search Patterns via MCS

In this appendix, we give a detailed account of the technique that we follow to generate the candidate search patterns. The technique was presented in previous work (Bernardini et al., 2017), but we repeat the material here to make this paper self-contained.

In S&T, the optimization solver's role is to select a set of search patterns and sequence them over time. To operate effectively, we need to provide the solver with an initial pool of *candidate search patterns* from which it chooses a subset to execute. It is preferable to keep the cardinality of this set small so as to reduce the computational complexity but also to have a large enough set of candidates to allow a high probability of successful tracking of the target to its destination. We perform MCS to identify points in the search area that present the highest probability of finding the target at different points in time and then create candidate patterns that have those points as their centres.

A.1 Graph Construction

We assume that the target is located in Euclidean 2-space and that this space is characterized by a *road network* (RN), where each road is a sequence of connected line segments. The target motion on each segment is assumed to follow a constant speed randomly and uniformly sampled in an interval $[\nu^{\min}, \nu^{\max}]$, where $\nu^{\min}$ and $\nu^{\max}$ are the minimum and maximum speed allowed in that segment depending on the road type. Each segment in a road is characterized by a *concealment* level $\eta \in [0, 1]$ that represents an estimate of how easy is for the target to hide from the observer when travelling over that segment.

We take a circle centred on the target's last known position (LKP) as the search area and then superimpose a *grid* $\mathcal{X}$ on it, with the side of each square cell being δ. To represent the topology of the search area, we build a graph $\mathcal{G} = \langle \mathrm{V}, \mathrm{E} \rangle$ based on the RN enclosed in the grid. V represents the set of cells that contain at least one road segment within the grid. Edges in E are those pairs (v, w) where v and w are adjacent cells in the grid and there exists a road segment that intersects both of them. Each edge (v, w) is labelled with: (i) the minimum $\nu^{\min}_{(v,w)}$ and maximum $\nu^{\max}_{(v,w)}$ speed allowed in the segment that connects v to w and (ii) its concealment level $\eta_{(v,w)}$. We denote by v_0 the cell that corresponds to the target LKP and assume that a set of target possible destinations, which we will identify with a subset $\mathcal{D} \subset \mathrm{V}$ of cells, is given together with a probability distribution (*PD*) over them: $\mu : \mathcal{D} \to [0, 1]$.

A.2 Probabilistic Motion Model

Given the graph $\mathcal{G}$, we define the *weight* of an edge (v, w) as $w_{(v,w)} := (\delta/\nu^{\max}_{(v,w)}(1-\alpha\eta_{(v,w)}))$, where the parameter $\alpha \in [0, 1]$ needs to be established case-by-case based on the desired trade-off between the time to travel an edge and the concealment level over it. Given a path γ in $\mathcal{G}$, we define the *cost* of γ as $cost(\gamma) := \sum_{(v_i,w_i)\in\gamma} w_{(v_i,w_i)}$.

From the graph $\mathcal{G}$ and for l equally partitioned values of α, we calculate the k cheapest loop-less paths from v_0 to each destination in $\mathcal{D}$ by using a variant of the Dijkstra's single-source-shortest-path algorithm (Yen, 1971). Given v_0 and a destination $x \in \mathcal{D}$, we denote with $\Gamma_x = \{\gamma_1, \ldots, \gamma_{(k \cdot l)}\}$ the set of the $(k \cdot l)$ cheapest paths associated with destination x. For each destination x, we define a *PD* over Γ_x, $\theta : \Gamma_x \to [0, 1]$ as follows: $\theta(\gamma) = \frac{1}{Z(\beta)} e^{-\beta cost(\gamma)}$, where $\beta \in \mathbb{R}$ is a free parameter and the normalising constant $Z(\beta)$ is the partition function $\sum_{\gamma_i \in \Gamma_x} e^{-\beta cost(\gamma_i)}$. When $\beta = 0$, the probability is uniform over all paths; when β increases, the cheapest path progressively becomes the most probable. This function, therefore, gives us the flexibility to treat different degrees of evasiveness within the same framework.

Given the graph $\mathcal{G}$, consider the subgraph $\mathcal{G}'$ determined by the LKP node v_0, the destination nodes $\mathcal{D}$ and the nodes on the cheapest paths $\Gamma_{x_1}, \ldots \Gamma_{x_d}$ that connect v_0 to the destination $x_1, \ldots, x_d$. Given a node w in this graph, consider the subgraph $\mathcal{G}_w$ that is determined by all the paths from w to the destination nodes $\mathcal{D}$ (these are subpaths of the paths in $\Gamma_{x_1} \cup \ldots \cup \Gamma_{x_d}$). We call $\Gamma(w)$ the set of paths $\gamma \in \mathcal{G}_w$ and we say that γ are *compatible* with w.

The target motion is modelled as a continuous time stochastic process $X(t)$ that takes values on V and is described as follows:

- the final destination cell $x \in \mathcal{D}$ is sampled according to the *PD* μ;
- the path $\gamma \in \Gamma_x$ from v_0 to x is sampled according to the *PD* θ;
- $X(t)$ moves with a velocity scaled by a constant factor ω uniformly sampled in the interval [0,1];

- $X(t)$ moves from v_0 to x by following the path $\gamma = (v_0, v_1, \ldots, v_l = x)$ and by jumping from v_k to v_{k+1} at the time t_k; and
- the jumping time t_k's are iteratively determined according to the following formula: $t_{k+1} - t_k = \delta/\nu_k$, where $\nu_k = \nu^{\min}_{(v_k,v_{k+1})} + \omega(\nu^{\max}_{(v_k,v_{k+1})} - \nu^{\min}_{(v_k,v_{k+1})})$.

A.3 Approximation of Marginal Distributions

$X(t)$ is a continuous time process, but we look at it only at certain time points. Given the mission time interval $[0, \mathrm{T}]$, we establish the time check points $t_0 = 0, t_1, \ldots, t_n$, where $t_{i+1} = t_i + \mathrm{T}/n$. Our goal is to estimate the marginal *PD* of the process $X(t)$ on the above check points. We then use these marginals to generate candidate search patterns.

Estimation of the marginal is performed through standard Monte Carlo Simulation (MCS). More specifically, we consider a set of M particles moving in the graph as independent realisations of the stochastic process $X(t)$. Let $\chi_j(t)$ be the position of the j^{th} particle at time t. We define the approximated distribution of the process $X(t)$ at time t_k as $q_v^{t_k} = |\ \{j|\chi_j(t_k) = v\}\ |\ /M$ for $v \in \mathrm{V}$. From the law of large numbers, we know that $q_v^{t_k}$ approximates, for a sufficiently large M, the true marginal distributions of $X(t_k)$.

A.4 Generation of Search Patterns

For each time check point t_i, we select the n nodes that have collected the highest number of particles and then generate n non-overlapping candidate search patterns centred around them, which are subsets of $\mathbb{R}^2$. We denote the set of all search patterns chosen at any time check point by $\mathcal{C}$. Each search pattern $\sigma \in \mathcal{C}$ has a time window $[t_\sigma^-, t_\sigma^+]$ associated with it that corresponds to the activation window of the pattern. This window is set up by calculating the shortest and longest time of arrival for the target to the pattern's centre. For every $\sigma \in \mathcal{C}$, we call V_σ the set of nodes in the graph $\mathcal{G}$ that are contained by σ (in the embedding environment $\mathbb{R}^2$), i.e. $V_\sigma = \{v \in V | v \subseteq \sigma\}$. We indicate with $\Gamma_\sigma = \bigcup_{v \in V_\sigma} \Gamma(v)$ the paths γ that are compatible with σ.

Appendix B. Iterative Update of Probabilities

Following Bernardini et al. (2016), we report here the derivation of the total probability of rediscovering the target after the execution of a sequence of search patterns Σ.

Let $\mathcal{F}_\sigma$ represent the event of finding the target in a search of the area covered by the pattern σ and $\tilde{\mathcal{F}}_\sigma$ its negation. Given a set of search patterns $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$, we call $\mathcal{F}_\Sigma$ the event of finding the target during the execution of the set Σ, and $\tilde{\mathcal{F}}_\Sigma$ its negation, i.e.:

$$\mathcal{F}_\Sigma = \mathcal{F}_{\sigma_1} \cup \cdots \cup \mathcal{F}_{\sigma_k} \qquad \tilde{\mathcal{F}}_\Sigma = \tilde{\mathcal{F}}_{\sigma_1} \cap \cdots \cap \tilde{\mathcal{F}}_{\sigma_k}$$

We call $P_\Sigma(\gamma)$ the probability that the target is following the path γ conditioned to the failure of the search patterns in Σ.

$$P_\Sigma(\gamma) \coloneqq \mathbb{P}(\text{target} \to \gamma | \tilde{\mathcal{F}}_\Sigma)$$

Given an initial probability distribution for each path, we have that $P_\emptyset(\gamma) = PD(\gamma)$.

The probability $P(\Sigma)$ of finding the target by executing the set of search patterns Σ is defined as:

$$P(\Sigma) \coloneqq \mathbb{P}(\mathcal{F}_\Sigma) \tag{30}$$

We now show how to calculate $P(\Sigma)$ in a recursive fashion. We start by analyzing the path probabilities $P_\Sigma(\gamma)$, which play a pivotal role in our computation. Given a set of search patterns Σ and a search pattern candidate $\sigma \in \tilde{\mathcal{C}} \setminus \Sigma$, we can write $P_\Sigma(\gamma)$ using a total probability argument:

$$P_\Sigma(\gamma) = P_{\Sigma\cup\{\sigma\}}(\gamma)\mathbb{P}(\tilde{\mathcal{F}}_\sigma|\tilde{\mathcal{F}}_\Sigma) + \mathbb{P}(\text{target} \to \gamma|\tilde{\mathcal{F}}_\Sigma \cap \mathcal{F}_\sigma)\mathbb{P}(\mathcal{F}_\sigma|\tilde{\mathcal{F}}_\Sigma) \tag{31}$$

The different terms in this equation can be rewritten as follows. We let $P_{\Sigma\cup\sigma*} \coloneqq \mathbb{P}(\mathcal{F}_\sigma|\tilde{\mathcal{F}}_\Sigma)$ and call $P_{\Sigma\cup\sigma*}$ the probability that the target is found during the execution of a search pattern σ, conditioned to the event that it has not been discovered earlier. This probability is the product of two terms: 1. the probability that the target is following a path compatible with σ (i.e. $\gamma \in \Gamma_\sigma$) computed according to the distribution $P_\Sigma(\gamma)$, which encodes the fact that the target has not been discovered earlier; and 2. the probability that the observer finds the target when it is in view, i.e. the detection probability ϕ_σ. In formula:

$$P_{\Sigma\cup\sigma*} = \phi_\sigma \sum_{\gamma\in\Gamma_\sigma} P_\Sigma(\gamma) \tag{32}$$

Let us consider now the term $\mathbb{P}(\text{target} \to \gamma|\tilde{\mathcal{F}}_\Sigma \cap \mathcal{F}_\sigma)$. To expand it further, we need to distinguish whether the path γ is in the set of destinations compatible with the pattern σ or not. Evidently, if $\gamma \notin \Gamma_\sigma$, this term is equal to 0. If $\gamma \in \Gamma_\sigma$, this term can be computed by simply conditioning the probability distribution $P_\Sigma(\gamma)$ on the subset of the destinations Γ_σ which are compatible with σ. We thus obtain the following expression:

$$\mathbb{P}(\text{target} \to \gamma|\tilde{\mathcal{F}}_\Sigma \cap \mathcal{F}_\sigma) = \begin{cases} \frac{P_\Sigma(\gamma)}{\sum\limits_{\eta\in\Gamma_\sigma} P_\Sigma(\eta)} & \text{if } \gamma \in \Gamma\sigma \\ \\ 0 & \text{if } x \notin \Gamma\sigma \end{cases} \tag{33}$$

If we substitute Equations (32) and (33) into (31), we obtain the following recursive structure for the computation of $P_{\Sigma\cup\{\sigma\}}(\gamma)$:

$$P_{\Sigma\cup\{\sigma\}}(\gamma) = \frac{P_\Sigma(\gamma) \cdot (1 - \phi_\sigma \cdot \mathbb{1}_{\Gamma_\sigma}(\gamma))}{1 - P_{\Sigma\cup\{\sigma\}*}} \tag{34}$$

where $\mathbb{1}$ is the indicator function: $\mathbb{1}_A(x) = 1$ if $x \in A$ and 0 otherwise.

A recursive structure for the computation of $P(\Sigma \cup \{\sigma\})$ can now be obtained as follows:

$$P(\Sigma \cup \{\sigma\}) = \mathbb{P}(\mathcal{F}_\Sigma) + \mathbb{P}(\mathcal{F}_\sigma|\tilde{\mathcal{F}}_\Sigma) \cdot \mathbb{P}(\tilde{\mathcal{F}}_\Sigma) \tag{35}$$

or, in more compact notation,

$$\begin{aligned} &P(\Sigma \cup \{\sigma\}) = P(\Sigma) + P_{\Sigma\cup\{\sigma\}*} \cdot (1 - P(\Sigma)) \\ &P(\emptyset) = 0 \end{aligned} \tag{36}$$

If we combine Equations (32), (33) and (36), we obtain an exact recursive formula for the computation of $P(\Sigma)$.

Appendix C. PDDL2.1 Specification of the Search Domain

```
(define (domain UAV)
    (:requirements :typing :durative-actions :fluents
     :timed-initial-literals :equality)
    (:types pattern waypoint destination uav)

    (:predicates
        (at ?u - uav ?p - waypoint)
        (active ?p - pattern)
        (beginAt ?w - waypoint  ?p - pattern)
        (endAt ?w - waypoint ?p - pattern))

    (:functions
        (prob ?d - destination)
        (previous-prob ?d - destination)
        (total-prob)
        (heuristic-approximation)
        (previous-total-prob
        (is-doing ?p - pattern ?u - uav)
        (is-uav ?u - uav)
        (timefor ?p - pattern)
        (distance ?p1 ?p2 - waypoint)
        (n-pattern)
        (n-pattern-active ?p - pattern))

    (:durative-action fly
        :parameters (?from ?tto - waypoint ?u - uav )
        :duration (= ?duration (distance ?from ?tto))
        :condition (and
            (at start (at ?u ?from))
            (at start (not (= ?from ?tto))))
        :effect (and
            (at start (not (at ?u ?from)))
            (at end (at ?u ?tto))))

    (:durative-action doPattern
        :parameters (?from ?to - waypoint ?p - pattern ?u - uav)
        :duration (=?duration (timefor ?p))
        :condition (and
            (at start (beginAt ?from ?p))
            (at start (endAt ?to ?p))
            (at start (at ?u ?from))
            (at start (active ?p))
            (at start (< (is-uav ?u) 10)))
        :effect (and
            (at end (at ?u ?to))
            (at start (not (at ?u ?from)))
            (forall (?pp - pattern)
                (and (at start (assign (is-doing ?pp ?u) 0))))
            (forall (?v - uav) (and (at start (assign (is-uav ?v) 0))))
            (at end (assign (is-doing ?p ?u) 1))
            (at end (assign (is-uav ?u) 1))
            (forall (?d - destination)
                (and (at end (assign (previous-prob ?d) (prob?d)))))
            (at end (assign (previous-total-prob) (total-prob)))
            (forall (?d - destination)
                (and (at end (increase (probability ?d) 0.0001))))
            (at end (increase (total-prob) (heuristic-approximation)))
            (at end (assign (n-pattern) (n-pattern-active ?p))))))
```

References

Abi-Zeid, I., Nilo, O., & Lamontagne, L. (2011). Resource allocation algorithms for planning search and rescue operations. *INFOR*, *49*(1), 15–30.

Adams, S. M., & Friedland, C. J. (2011). A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management. In *Proceedings of the 9th International Workshop on Remote Sensing for Disaster Response*, Vol. 8.

Alaei, S., & Malekian, A. (2010). Maximizing sequence-submodular functions and its application to online advertising. In *arXiv:1009.4153v1*, pp. 1–18.

Alon, N., Gamzu, I., & Tennenholtz, M. (2012). Optimizing budget allocation among channels and influencers. In *Proceedings of the 21st international conference on World Wide Web*, pp. 381–388. ACM.

Ammad-Udin, M., Mansour, A., Le Jeune, D., Aggoune, E. H. M., & Ayaz, M. (2016). Uav routing protocol for crop health management. In *Proceedings of the 24th European Signal Processing Conference (EUSIPCO)*, pp. 1818–1822. IEEE.

Baier, J. A., Bacchus, F., & McIlraith, S. A. (2009). A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, *173*(5), 593.

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *Constraint-based Scheduling.* Kluwer Academic Publishers.

Beck, J. C., Davenport, A. J., Davis, E. D., & Fox, M. S. (1998). The ODO project: Toward a unified basis for constraint-directed scheduling. *Journal of Scheduling*, *1*(2), 89–125.

Beck, J. C. (2007). Solution-guided multi-point constructive search for job shop scheduling. *Journal of Artificial Intelligence Research*, *29*, 49–77.

Beck, J. C. (2010). Checking-up on branch-and-check. In Cohen, D. (Ed.), *Proceedings of the 16th International Conference on the Principles and Practice of Constraint Programming (CP)*, pp. 84–98, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bellingham, J. S., Tillerson, M., Alighanbari, M., & How, J. P. (2002). Cooperative path planning for multiple uavs in dynamic and uncertain environments. In *Proceedings of the 41st IEEE Conference on Decision and Control*, Vol. 3, pp. 2816–2822 vol.3.

Benkoski, S. J., Monticino, M. G., & Weisinger, J. R. (1991). A survey of the search theory literature. *Naval Research Logistics*, *38*(4), 469–494.

Benton, J., Coles, A. J., & Coles, A. (2012). Temporal planning with preferences and time-dependent continuous costs.. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, Vol. 77, p. 78.

Bernardini, S., Fox, M., & Long, D. (2017). Combining temporal planning with probabilistic reasoning for autonomous surveillance missions. *Autonomous Robots*, *41*(1), 181–203.

Bernardini, S., Fox, M., Long, D., & Bookless, J. (2013). Autonomous search and tracking via temporal planning. In *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS)*.

Bernardini, S., Fox, M., Long, D., & Piacentini, C. (2016). Leveraging probabilistic reasoning in deterministic planning for large-scale autonomous search-and-tracking. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*.

Bernardini, S., Fox, M., Long, D., & Piacentini, C. (2017a). Boosting search guidance in problems with semantic attachments. In *Proceedings of the 27h International Conference on Automated Planning and Scheduling, (ICAPS)*, pp. 29–37.

Bernardini, S., Fox, M., Long, D., & Piacentini, C. (2017b). Deterministic vs probabilistic methods for searching for an evasive target. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*.

Bertuccelli, L. F., & How, J. P. (2006). Uav search for dynamic targets with uncertain motion models. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pp. 5941–5946.

Bondi, E., Fang, F., Hamilton, M., Kar, D., Dmello, D., Choi, J., Hannaford, R., Iyer, A., Joppa, L., & Tambe, M. (2018). Spot poachers in action: Augmenting conservation drones with automatic detection in near real time. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Booth, K. E. C., Do, M., Beck, J. C., Rieffel, E. G., Venturelli, D., & Frank, J. (2018). Comparing and integrating constraint programming and temporal planning for quantum circuit compilation. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling, (ICAPS)*, pp. 366–374.

Booth, K. E. C., Nejat, G., & Beck, J. C. (2016a). A constraint programming approach to multi-robot task allocation and scheduling in retirement homes. In *Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 539–555.

Booth, K. E. C., Tran, T. T., Nejat, G., & Beck, J. C. (2016b). Mixed-integer and constraint programming techniques for mobile robot task planning. *IEEE Robotics and Automation Letters*, *1*(1), 500–507.

Bourgault, F., Furukawa, T., & Durrant-Whyte, H. F. (2004). Process model, constraints, and the coordinated search strategy. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA))*, pp. 5256–5261.

Bourgault, F., Furukawa, T., & Durrant-Whyte, H. F. (2006). Optimal search for a lost target in a bayesian world. In *Field and Service Robotics*, Vol. 24 of *Springer Tracts in Advanced Robotics*, pp. 209–222. Springer Berlin.

Calinescu, G., Chekuri, C., Pál, M., & Vondrák, J. (2007). Maximizing a submodular set function subject to a matroid constraint. In *International Conference on Integer Programming and Combinatorial Optimization*, pp. 182–196. Springer.

Carvalho, C., Teichteil-Konigsbuch, F., & Lesire, C. (2013). Multi-target detection and recognition by uavs using online pomdps. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*.

Charrow, B., Kumar, V., & Michael, N. (2014). Approximate representations for multi-robot control policies that maximize mutual information. *Autonomous Robots*, *37*(4), 383–400.

Chen, W., Chen, Y., & Weinberger, K. (2015). Filtered search for submodular maximization with controllable approximation bounds. In *Eighteenth International Conference on Artificial Intelligence and Statistics*, Vol. 38, pp. 156–164.

Chmaj, G., & Selvaraj, H. (2015). Distributed processing applications for uav/drones: a survey. In *Progress in Systems Engineering*, pp. 449–454. Springer.

Chung, C. F., & Furukawa, T. (2006). Coordinated search-and-capture using particle filters. In *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1–6.

Chung, T. H., & Carpin, S. (2011). Multiscale search using probabilistic quadtrees. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2546–2553.

Coles, A. J., Coles, A. I., Fox, M., & Long, D. (2009). Temporal planning in domains with linear processes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1671–1676.

Coles, A. J., Coles, A. I., Fox, M., & Long, D. (2010). Forward-chaining partial-order planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*.

Coles, A., Fox, M., Halsey, K., Long, D., & Smith, A. (2009). Managing concurrency in temporal planning using planner-scheduler interaction. *Artificial Intelligence*, *173*(1), 1–44.

Dechter, R., Meiri, I., & Pearl, J. (1991). Temporal constraint networks. *Artificial Intelligence*, *49*, 61–95.

Domshlak, C., & Mirkis, V. (2015). Deterministic oversubscription planning as heuristic search: Abstractions and reformulations. *Journal of Artificial Intelligence Research*, *52*, 97–169.

Doran, J. E., & Michie, D. (1966). Experiments with the graph traverser program. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *294*(1437), 235–259.

Enright, J. J., Savla, K., Frazzoli, E., & Bullo, F. (2009). Stochastic and dynamic routing problems for multiple uninhabited aerial vehicles. *Journal of Guidance, Control, and Dynamics*, *32*(4), 1152–1166.

Fazel-Zarandi, M. M., Berman, O., & Beck, J. C. (2013). Solving a stochastic facility location/fleet management problem with logic-based benders' decomposition. *IIE Transactions*, *45*(8), 896–911.

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, *20*.

Furukawa, T., Bourgault, F., Lavis, B., & Durrant-Whyte, H. F. (2006). Recursive bayesian search-and-tracking using coordinated UAVs for lost targets. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2521–2526.

Furukawa, T., Durrant-Whyte, H. F., & Lavis, B. (2007). The element-based method — theory and its application to bayesian search and tracking. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2807–2812.

Furukawa, T., Mak, L. C., Durrant-Whyte, H. F., & Madhavan, R. (2012). Autonomous bayesian search and tracking, and its experimental validation. *Advanced Robotics*, *26*(5-6), 461–485.

Gerevini, A. E., Haslum, P., Long, D., Saetti, A., & Dimopoulos, Y. (2009). Deterministic planning in the fifth international planning competition: Pddl3 and experimental evaluation of the planners. *Artificial Intelligence*, *173*(5-6), 619–668.

Girard, A. R., Howell, A. S., & Hedrick, J. K. (2004). Border patrol and surveillance missions using multiple unmanned air vehicles. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Vol. 1, pp. 620–625. IEEE.

Harjunkoski, I., & Grossmann, I. E. (2002). Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering*, *26*(11), 1533–1552.

Haslum, P., Ivankovic, F., Ramirez, M., Gordon, D., Thiébaux, S., Shivashankar, V., & Nau, D. S. (2018). Extending classical planning with state constraints: Heuristics and search for optimal planning. *Journal of Artificial Intelligence Research*, *62*, 373–431.

He, R., Bachrach, A., & Roy, N. (2010). Efficient planning under uncertainty for a target-tracking micro-aerial vehicle. In *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8.

Hodgson, J. C., Baylis, S. M., Mott, R., Herrod, A., & Clarke, R. H. (2016). Precision wildlife monitoring using unmanned aerial vehicles. *Scientific reports*, *6*, 22574.

Hoffmann, G. M., & Tomlin, C. J. (2010). Mobile sensor network control using mutual information methods and particle filters. *IEEE Transactions on Automatic Control*, *55*(1), 32–47.

Hollinger, G., Kehagias, A., & Singh, S. (2007). Probabilistic strategies for pursuit in cluttered environments with multiple robots. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3870–3876.

Hollinger, G. (2010). Search in the physical world. Tech. rep., Robotics Institute, Carnegie Mellon University.

Hooker, J., & Ottosson, G. (2003). Logic-based benders decomposition. *Mathematical Programming*, *96*(1), 33–60.

Hsu, D., Sun, W., & Rong, L. N. (2008). A point-based pomdp planner for target tracking. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2644–2650.

Jawaid, S. T., & Smith, S. L. (2015). Submodularity and greedy algorithms in sensor scheduling for linear dynamical systems. *Automatica*, *61*, 282–288.

Karich, P. (2015). GraphHopper. https://graphhopper.com. Accessed: 2016-09-11.

Kelareva, E., Tierney, K., & Kilby, P. (2013). Cp methods for scheduling and routing with time-dependent task costs. In *Proceedings of the 10th International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems (CPAIOR)*, pp. 111–127. Springer.

Koopman, B. (1946). Search and screening – operations research evaluation group report 56. Tech. rep., Center for Naval Analyses.

Kratzke, T., Stone, L., & Frost, J. (2010). Search and rescue optimal planning system. In *Proceedings of the 13th Conference on Information Fusion*, pp. 1–8.

Krause, A., & Guestrin, C. (2011). Submodularity and its applications in optimized information gathering. *ACM Transactions on Intelligent Systems and Technology*, *2*(4), 1–20.

Kulik, A., Shachnai, H., & Tamir, T. (2009). Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the 20th annual ACM-SIAM symposium on Discrete algorithms*, pp. 545–554. Society for Industrial and Applied Mathematics.

Laborie, P. (2009). IBM ILOG CP optimizer for detailed scheduling illustrated on three problems. In *Proceedings of the 6th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR)*, pp. 148–162.

Laborie, P., & Messaoudi, B. (2017). New results for the GEO-CAPE observation scheduling problem. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 382–390.

Laborie, P., Refalo, P., & Shaw, P. (2013). Model presolve, warmstart and conflict refining in cp optimizer. In *Presentation at the CP2013 Workshop on CP Solvers: Modeling, Applications, Integration, and Standardization.*

Laborie, P., & Rogerie, J. (2008). Reasoning with conditional time-intervals.. In *Proceedings of the FLAIRS Conference*, pp. 555–560.

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling. *Constraints*, *23*(2), 210–250.

Lau, H., Huang, S., & Dissanayake, G. (2005). Optimal search for multiple targets in a built environment. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Lau, H., Huang, S., & Dissanayake, G. (2006). Probabilistic search for a moving target in an indoor environment. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Lavis, B., & Furukawa, T. (2008). HyPE: Hybrid particle-element approach for recursive bayesian searching and tracking. In *Proceedings of the 2008 Robotics: Science and Systems Conference*, pp. 135–142.

Lin, H., & Bilmes, J. (2010). Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 912–920. Association for Computational Linguistics.

Lin, L., & Goodrich, M. A. (2014). Hierarchical heuristic search using a gaussian mixture model for uav coverage planning. *IEEE Transactions on Cybernetics*, *44*, 2532–2544.

Liu, P., Chen, A. Y., Huang, Y.-N., Han, J.-Y., Lai, J.-S., Kang, S.-C., Wu, T.-H., Wen, M.-C., & Tsai, M.-H. (2014). A review of rotorcraft unmanned aerial vehicle (UAV) developments and applications in civil engineering. *Smart Structures Systems*, *13*(6), 1065–1094.

Min, H., Jayaraman, V., & Srivastava, R. (1998). Combined location-routing problems: A synthesis and future research directions. *European Journal of Operational Research*, *108*(1), 1 – 15.

Minoux, M. (1978). Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, *XXXIII*(2), 234–243.

Morin, M., Abi-Zeid, I., Quimper, C.-G., & Nilo, O. (2017). Decision support for search and rescue response planning. In *Proceedings of the 14th ISCRAM Conference*.

Nemhauser, G. L., & Wolsey, L. (1978). An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, *14*, 265–294.

Parambath, S. P., Vijayakumar, N., & Chawla, S. (2018). SAGA: A submodular greedy algorithm for group recommendation. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Paterson, J. G., Timmons, E., & Williams, B. C. (2014). A scheduler for actions with iterated durations. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*.

Piacentini, C., Alimisis, V., Fox, M., & Long, D. (2015). An extension of metric temporal planning with application to AC voltage control. *Artificial Intelligence*, *229*, 210–245.

Pitre, R. R., Li, X. R., & Delbalzo, R. (2012). UAV route planning for joint search and track missions: An information-value approach. *IEEE Transactions on Aerospace and Electronic Systems*, *48*(3), 2551–2565.

Pommerening, F. (2017). New perspectives on cost partitioning for optimal classical planning. Tech. rep., Faculty of Science, University of Basel.

Richardson, H. R., & Discenza, J. H. (1980). The united states coast guard computer-assisted search planning system (CASP). *Naval Research Logistics Quarterly*, *27*(4), 659–680.

Richardson, H. R., & Stone, L. D. (1971). Operations analysis during the underwater search for Scorpion. *Naval Research Logistics*, *18*, 141–157.

Rodriguez, J. (2007). A constraint programming model for real-time train scheduling at junctions. *Transportation Research Part B: Methodological*, *41*(2), 231–245.

Roy, N., & Earnest, C. (2006). Dynamic action spaces for information gain maximization in search and exploration. In *2006 American Control Conference*.

Ryan, A., & Hedrick, J. K. (2010). Particle filter based information-theoretic active sensing. *Robotics and Autonomous Systems*, *58*(5), 574 – 584.

Sakaue, S., & Ishihata, M. (2018). Accelerated best-first search with upper-bound computation for submodular function maximization.. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*.

Sewell, E. C., & Jacobson, S. H. (2012). A branch, bound, and remember algorithm for the simple assembly line balancing problem. *INFORMS Journal on Computing*, *24*(3), 433–442.

Smith, D. E., & Weld, D. S. (1997). Temporal planning with mutual exclusion reasoning. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 326–333.

Stern, R. T., Kiesel, S., Puzis, R., Felner, A., & Ruml, W. (2014). Max is more than min: Solving maximization problems with heuristic search. In *Proceedings of the 7th Annual Symposium on Combinatorial Search (SOCS)*, pp. 148–156.

Stone, L. D. (1975). *The Theory of Optimal Search*. Operations Research Society of America.

Sun, X., Uras, T., Koenig, S., & Yeoh, W. (2012). Incremental ARA*: An incremental anytime search algorithm for moving-target search. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*.

Tian, X., Bar-Shalom, Y., & Pattipati, K. R. (2008). Multi-step look-ahead policy for autonomous cooperative surveillance by UAVs in hostile environments. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pp. 2438–2443.

Tierney, K., Coles, A. J., Coles, A., Kroer, C., Britt, A. M., & Jensen, R. M. (2012). Automated planning for liner shipping fleet repositioning.. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 279–287.

Tisdale, J., Ryan, A., Kim, Z., Tornqvist, D., & Hedrick, J. (2008). A multiple uav system for vision-based search and localization. In *Proceedings of the 2008 American Control Conference*, pp. 1985–1990.

Tran, T. T., Vaquero, T. S., Nejat, G., & Beck, J. C. (2017). Robots in retirement homes: Applying off-the-shelf planning and scheduling to a team of assistive robots. *Journal of Artificial Intelligence Research*, *58*, 523–590.

Trummel, K. E., & Weisinger, J. R. (1986). The complexity of the optimal searcher path problem. *Operations Research*, *34*(2), 324–327.

Vidal, R., Shakernia, O., Kim, H., Shim, D., & Sastry, S. (2002). Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation. *IEEE Trans. Robotics and Automation*, *18*(5), 662?669.

Weil, G., Heus, K., Francois, P., & Poujade, M. (1995). Constraint programming for nurse scheduling. *IEEE Engineering in medicine and biology magazine*, *14*(4), 417–422.

White, B. A., Tsourdos, A., Ashokaraj, I., Subchan, S., & Zbikowski, R. (2008). Contaminant cloud boundary monitoring using network of UAV sensors. *IEEE Sensors Journal*, *8*(10), 1681–1692.

Yen, J. Y. (1971). Finding the K shortest loopless paths in a network. *Management Science*, *44*, 712–716.