Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Leveraging Probabilistic Reasoning in Deterministic Planning for Large-Scale Autonomous Search-and-Tracking

Sara Bernardini
Department of Computer Science
Royal Holloway, University of London
Egham, Surrey, UK, TW20 0EX
sara.bernardini@rhul.ac.uk

Maria Fox, Derek Long, Chiara Piacentini
Department of Informatics
King's College London
London, UK, WC2R 2LS
firstname.lastname@kcl.ac.uk

Abstract

Search-And-Tracking (SaT) is the problem of searching for a mobile target and tracking it once it is found. Since SaT platforms face many sources of uncertainty and operational constraints, progress in the field has been restricted to simple and unrealistic scenarios. In this paper, we propose a new *hybrid* approach to SaT that allows us to successfully address large-scale and complex SaT missions. The probabilistic structure of SaT is compiled into a deterministic planning model and Bayesian inference is directly incorporated in the planning mechanism. Thanks to this tight integration between automated planning and probabilistic reasoning, we are able to exploit the power of both approaches. Planning provides the tools to efficiently explore big search spaces, while Bayesian inference, by readily combining prior knowledge with observable data, allows the planner to make more informed and effective decisions. We offer experimental evidence of the potential of our approach.

1 Introduction

In a Search-And-Tracking (SaT) mission, a searching vehicle, the observer, wishes to locate the position of a moving object, the target, and to track it to destination upon finding it. Examples of SaT missions are a UAV searching for life-rafts drifting with current, a police helicopter tracking a suspected criminal over a road network and a small drone escorting a worker who performs risky tasks in a factory.

SaT is an important component of many autonomous surveillance and Search-and-Rescue (SaR) operations, but it remains challenging due to the uncertainty inherent in it: the target location is unknown and moving, its motion model is often indeterminate and detection capabilities are imperfect. In addition, search platforms have limited resources, e.g. fuel or battery, and face many operational constraints, e.g., search paths must be equally spaced and parallel. Despite these constraints, the observer needs to make decisions quickly as it operates in time-critical situations.

Efficient solutions to SaT have been proposed under restrictive simplifying assumptions such as the search area being small, the temporal horizon being short and the target's motion model being simple (e.g., targets being stationary or in Markovian motion) (Stone 1975; Bourgault, Furukawa, and Durrant-Whyte 2006; Lavis and Furukawa 2008; He, Bachrach, and Roy 2010; Lin and Goodrich 2014). Several of these solutions are based on Recursive Bayesian Estimation (RBE) and a greedy search over a very short planning horizon (typically, a one-step lookahead). Although this purely probabilistic approach is successful for small-scale and simple SaT problems, it fails in the face of all the constraints that characterise real-world SaT operations. When the geographical area of search is large (hundreds of square km) and heterogeneous, the temporal horizon is long (hours) and the target moves in an unpredictable way according to its own intentions, probabilistic techniques become computationally too expensive.

To control complexity, Bernardini et al. (2013) reformulate SaT from a continuous optimisation problem into a combinatorial search problem by discretising it in time and space. SaT is modelled as a planning task consisting of deciding where to search and which search manoeuvres to perform in order to maximise the likelihood of recovering the target. This problem is then solved by using an off-the-shelf high-performing planner, OPTIC (Benton, Coles, and Coles 2012). In a radical departure from RBE techniques, the probabilistic aspects of the search problem are compiled away and a deterministic formulation is used to solve it. Plan-based SaT may find good policies for large-scale problems with predictable target behaviours, but it struggles with serious inaccuracies when the target acts in a more sophisticated way by neglecting its physical motion in the environment and the outcome of previous searches. In (Bernardini, Fox, and Long 2015), an improvement of plan-based SaT is presented, which is obtained by incorporating information about the target motion model in the generation of the manoeuvres that the planner considers for selection. This information is derived from running Monte Carlo Simulation outside the planner. Although the MCS-based approach is far more accurate than plan-based SaT, it is still not very effective over long temporal horizons since it does not take into account the information learned by the UAV during previous unsuccessful searches.

In this paper, we propose a new *hybrid* approach to SaT that combines *automated planning* with *Bayesian reasoning*. We offer a completely new formulation of the problem that captures the probabilistic motion model of the target within the planning domain and allows the planner to exploit Bayesian inference to make and update predictions on the target position over time based on the outcome of past

observations. In our approach, we benefit from the strengths of both planning and probabilistic reasoning: the former enables us to find efficient solutions quickly; the latter mitigates the intrinsic uncertainty of SaT missions by updating prior knowledge for new observations. Our model is very expressive and allows the planner to create robust plans that account for the target motion model, the result of previous searches and the detection capabilities of the sensors available to the UAV. Our experiments provide evidence that the integration of planning with Bayesian reasoning through this hybrid model opens the door to solving complex real-world SaT problems over long temporal horizons and large geographical spaces.

2 The Search Problem

Of the two phases of search and tracking, we focus on search in this paper. We assume that the target, a road vehicle, is located in the Euclidean 2-space and that this space is characterised by a *road network* (RN), where each road is a sequence of line segments. The target is in uniform motion on such segments and moves independently of the searcher's actions. It needs to reach a specific destination and chooses an efficient path to do so.

The observer, a fixed-wing UAV, flies a series of manoeuvres to locate the target. The UAV is equipped with an imaging system to scan the RN, which is susceptible to error from the features of the environment. We estimate the UAV search effort in terms of time and assume that the effort is constrained by the UAV's maximum endurance.

The search problem is concerned with finding a set of manoeuvres for the UAV that maximises the probability of detecting the target given the constraints on the effort.

3 A Hybrid Approach to Search

The UAV exploits standard *search patterns* (spirals and lawnmowers) to survey the search region and uses them as building blocks of a search plan that attempts to maximise the expectation of discovering the target. Using standard flight patterns is a sensible choice not only for scalability reasons, but also because it is line with SaT and SaR international guidelines (IMO 2013). The plan-based approach to SaT works in two stages: first, we create a set of *candidate* search patterns, which covers the most promising locations to discover the target, and then we feed this set into the planner, which is in charge of selecting a subset of candidates and sequencing them over time. The planner looks for plans that maximise the probability of finding the target while favouring manoeuvres that minimise the use of the UAV's resources (flight time).

We formulate SaT as a *deterministic* planning task despite the uncertainty inherent in it. We construct the plans entirely under the assumption that the target remains undiscovered, because, if the target is found, the plan is abandoned. Hence, we do not need to model the target's position explicitly in so removing the source of uncertainty of the problem.

To implement this approach to SaT, we need to manage two tasks: (i) identifying an initial pool of candidate search patterns from which the planner chooses the ones to execute; and (ii) identifying a mechanism to establish which sequence of patterns is best suited to discover the target.

To address the first task, we could create an arbitrarily large number of candidates to cover the whole area of operation. However, to reduce the computational complexity, it is preferable to keep the number small without compromising the chances of discovering the target. Building on the work by Bernardini, Fox, and Long (2015), we perform *Monte Carlo Simulation* (MCS) to identify points in the search area that present the highest probability of finding the target at different points in time. Then, we create candidate patterns centred around those points and that are active during the period in which the target can plausibly be there. Effectively, MCS provides us with a fine-grained time-dependent prior PD for the target location.

As for the second task, i.e. finding a sequence of search patterns that maximises the probability of finding the target, the planner needs to be equipped with a mechanism for making predictions about where the target is driving to as well as for updating such predictions over time in view of the target's motion and the results of the previous searches. This is when our model based on *Bayesian reasoning* comes into play by providing a powerful mechanism for combining previous knowledge with new observations. We compile the prior PD provided by the MCS as well as the target motion model into a planning task specification and prompt the planner to reason about possible combinations of patterns. The planner adds each new pattern to the plan under the assumption that the previous one has failed to discover the target. The failed pattern gives the planner *negative* information regarding the target's location, which the planner can take into account to update the current prior PD. As patterns roll on, the posterior probability of the locations in the vicinity of those that have already been searched become lower than the posterior of the locations that have not been explored yet. Upon completion of this iterative process, the planner is in a position to choose from a wide array of possible search sequences the one that carries the highest probability of finding the target and can dispatch that sequence to the observer. Figure 1 shows the different phases of a SaT mission and how we tackle them.

4 Compiling Probabilistic SaT Problems into Deterministic Planning Models

This section focuses on the theoretical framework behind our hybrid SaT model, Section 5 describes the calculations that are performed during planning and, finally, Section 6 illustrates how such a framework can be expressed in PDDL and run through a PDDL planner.

To construct the model, we assume that the target Last Known Position (LKP) is given as well as a set of target possible destinations, $\mathcal{D}$, with $d = |\mathcal{D}|$. We also assume that a PD over the destination is known and consider here a uniform PD.As already mentioned, we obtain candidate search patterns by running MCS. First, we determine an optimal area where the search should be deployed. We take a circular sector centred on the target LKP and extending out-

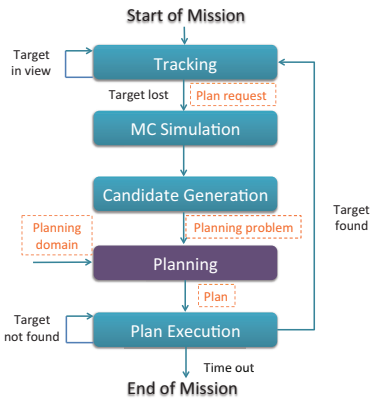

Figure 1: Phases of a plan-based SaT mission

wards with its symmetry axis aligned with the target's average bearing over the period it was observed. We then superimpose a grid on this sector and build a graph $\mathcal{G} = (C, E)$ based on the RN enclosed in the grid (C is the set of the cells in the grid and $(v, w) \in E$ if there is a road in the RN that connects the cells v and w). We use this graph to represents the topology of the search area. Since the nodes in C are cells in the grid, they can be seen as subsets of $\mathbb{R}^2$. This natural Euclidean embedding will be used throughout the paper. The set of cells corresponding to the target destinations will still be denoted by $\mathcal{D}$ and its elements, which we call destination nodes, by $x_1, x_2, \ldots, x_d$. We indicate the cell corresponding to the target LKP by $\bar{v} \in C$ and call it LKP node. From the graph $\mathcal{G}$, we calculate the shortest path from the LKP node to each destination node in $\mathcal{D}$ by using Dijkstra's single-source-shortest-path algorithm.

Given the graph $\mathcal{G}$, let us consider the subgraph determined by the LKP node $\bar{v}$, the destination nodes $\mathcal{D}$ and finally the nodes on the shortest paths that connect $\bar{v}$ to each $x_1, x_2, \ldots, x_d$ (we assume unique shortest paths, but multiple paths from the LKP node to the destination nodes can be easily incorporated in our model). This subgraph is a tree $\mathcal{T}$ with a set of vertices $V \subseteq C$ rooted in $\bar{v} \in V$. Given $w \in V$, we denote by $\mathcal{T}_w$ the subtree of $\mathcal{T}$ rooted in w. We denote by $\mathcal{D}(w)$ the set of destination nodes in $\mathcal{T}_w$ and call them *compatible* with w. If $W \subseteq V$, we put $\mathcal{D}(W) = \cup_{w \in W} \mathcal{D}(w)$.

We use the tree $\mathcal{T}$ to run MCS. We generate a set of particles, where each particle represents a guess of where the target might be located at any time point. We assign a destination to each particle by randomly choosing it based on the destination PD. The speed of the particle on each edge is randomly sampled from the PD of the target possible speeds for the type of road corresponding to the edge. Then, we simulate motion on the tree $\mathcal{T}$ assuming that each particle proceeds towards its destination without detours and travelling at a constant speed on each edge. Simulation is in continuous time and particles live both on nodes and edges.

Given the total mission time $\bar{t}$, we establish a set of time check points $t_0, \ldots, t_n$, where t_0 is the start of the mission and each $t_0 < \cdots < t_n < (t_0 + \bar{t})$. We simulate the motion of the particles up to each check point t_i and store their arrival node (we take the source node if the arrival position is on an edge) for each temporal slice. At the end of the simulation, we obtain an approximate representation of where the target might be located at the different time check points. The particles end up distributing over the main roads and, as time passes, clustering around the destinations.

For each time check point t_i, we select the *two* nodes that have collected the highest number of particles and then generate two candidate search patterns centred around them, which are subsets of $\mathbb{R}^2$. Choosing two patterns per time slice is clearly an approximation, which we use to keep complexity under control. However, our method can be easily generalised to create more patterns for each time slice. As for the type of patterns to use, we favour spirals in the part of the search area that is closest to the origin and in regions with a high density road, because they ensure precise coverage, while we use lawnmowers in rural areas or elongated stretches covering major roads [ref-removed].

We denote the set of all search patterns chosen at any time check point by Σ. Each search pattern $\sigma \in \Sigma$ 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 earliest and latest time of arrival for the target to the centre of the pattern. For every $\sigma \in \Sigma$, we denote by V_σ the set of nodes in the tree $\mathcal{T}$ that are contained by σ (in the embedding environment $\mathbb{R}^2$), i.e. $V_\sigma = \{v \in V | v \subseteq \sigma\}$. $\mathcal{D}_\sigma = \mathcal{D}(V_\sigma)$ indicates the set of destination nodes compatible with σ. A plan skeleton is a sequence of elements in Σ, $S = (\sigma_1, \sigma_2, \ldots, \sigma_{\bar{k}})$, with $\bar{k}$ indicating its length.

For each destination $x \in \mathcal{D}$, we call $P_S^{(k)}(x)$ the probability that the target is driving towards x at time step k after executing the patterns in S and provided that the searches in patterns $\sigma_1, \ldots, \sigma_k$ have failed. Initially, all the destinations are equally probable. However, in exploring the search area, we gain information about the intentions of the target and we can update the probabilities consequently. In a Bayesian fashion, if a search in σ has proven unsuccessful, we decrease the probabilities of the destinations compatible with σ since it now appears more unlikely that the target is heading towards one of them. Similarly, we increase the probabilities of the destinations that are incompatible with σ because the probability of finding the target there has now increased.

The probability of finding the target by executing a pattern σ depends not only on the presence of the target in the area covered by it, but also on the accuracy of the sensors used to scan such an area. We associate a *detection probability* to each pattern σ, which we call γ_σ. This is the probability of finding the target in an execution of the pattern σ conditioned to the target having initially chosen any of the destination nodes in $\mathcal{D}_\sigma$. The function γ_σ encodes both the randomness in the motion of the target and the sensor noise and is conditioned to the execution of its corresponding pattern σ within the associated time window.

Finally, we call $P(S)$ and $T(S)$, respectively, the probability and an approximation of the expected time (precisely defined in the next section) of finding the target by executing the plan skeleton S. We will consider an objective function of type $G(S) = P(S) - kT(S)$ where $k \geq 0$ is a con-

stant parameter. Maximisation of $G(S)$ leads to plans that balance, depending on k, high probability of discovery and time to complete the mission.

5 Iterative Update of Probabilities

To compute the objective function $G(S)$, we develop a formal framework based on *dynamic programming* for computing the probabilities associated with the destinations, the total probability and the expected time.

5.1 Notation

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 plan skeleton $S = (\sigma_1, \sigma_2, \ldots, \sigma_{\bar{k}})$, we define $\mathcal{F}_S^{(k)}$ as the event of finding the target at time step k by executing S and $\tilde{\mathcal{F}}_S^{(k)}$ its negation, i.e.:

$$\mathcal{F}_S^{(k)} = \mathcal{F}_{\sigma_1} \cup \cdots \cup \mathcal{F}_{\sigma_k} \qquad \tilde{\mathcal{F}}_S^{(k)} = \tilde{\mathcal{F}}_{\sigma_1} \cap \cdots \cap \tilde{\mathcal{F}}_{\sigma_k}$$

For each time step k, we associate a PD to each destination $x \in \mathcal{D}$, indicated as $P_S^{(k)}(x)$ and defined as follows:

$$P_S^{(k)}(x) = \mathbb{P}(\text{target} \rightarrow x | \tilde{\mathcal{F}}_S^{(k)}) \tag{1}$$

$P_S^{(k)}(x)$ is the probability at time step k that the target is driving to destination x provided that the searches in patterns $\sigma_1, \ldots, \sigma_k$ have failed. We have that $P_S^{(0)}(x) = \frac{1}{d}$ since we assume a uniform prior PD for the destinations.

The probability $P(S)$ of finding the target by executing the plan skeleton S and the probability $P^{(k)}(S)$ of finding the target within time step k are given by:

$$P(S) = \mathbb{P}(\mathcal{F}_S^{(\bar{k})}) \quad (2) \qquad P^{(k)}(S) = \mathbb{P}(\mathcal{F}_S^{(k)}) \quad (3)$$

The approximated expected time is defined by assuming that if the target is discovered by executing the pattern σ, the discovery time is given by the midpoint t_σ of the activation window $[t_\sigma^-, t_\sigma^+]$ ($t_\sigma = (t_\sigma^+ - t_\sigma^-)/2$). In formula:

$$T(S) := \sum_{j=1}^{\bar{k}} t_{\sigma_j} (P^{(j)}(S) - P^{(j-1)}(S))$$

In the next section, we calculate $P(S)$ and $T(S)$ in an iterative fashion.

5.2 Iterative Formulas for $P(S)$ and $T(S)$

We start by analysing the destination probabilities $P_S^{(k)}(x)$, which play a pivotal role in our computation. A recursive formula for $P_S^{(k)}(x)$ can be obtained by starting from $P_S^{(k-1)}(x)$ and using a total probability argument:

$$\begin{aligned} &P_S^{(k-1)}(x) = P_S^{(k)}(x) \cdot \mathbb{P}(\tilde{\mathcal{F}}_{\sigma_k} | \tilde{\mathcal{F}}_S^{(k-1)}) + \\ &\mathbb{P}(\text{target} \rightarrow x | \tilde{\mathcal{F}}_S^{(k-1)} \cap \mathcal{F}_{\sigma_k}) \cdot \mathbb{P}(\mathcal{F}_{\sigma_k} | \tilde{\mathcal{F}}_S^{(k-1)}) \end{aligned} \tag{4}$$

The different terms in this equation can be rewritten as follows. Let us put $P_{S*}^{(k-1)} := \mathbb{P}(\mathcal{F}_{\sigma_k} | \tilde{\mathcal{F}}_S^{(k-1)})$. This term represents the probability that the target is found during the execution of pattern σ_k at time step k conditioned to the event that it has not been discovered earlier, $\tilde{\mathcal{F}}_S^{(k-1)}$. It is thus the product of two terms: (i) the probability that the target has chosen any of the destinations compatible with σ_k (i.e. a destination in $\mathcal{D}_{\sigma_k}$) computed according to the distribution $P_S^{(k-1)}(x)$, which encodes the fact that the target has not been discovered earlier; and (ii) the probability that the observer finds the target when it is in view, i.e. the detection probability γ_{σ_k}.

In formula:

$$P_{S*}^{(k-1)} = \sum_{y \in \mathcal{D}_{\sigma_k}} P_S^{(k-1)}(y) \cdot \gamma_{\sigma_k} \tag{5}$$

Let us consider now the term $\mathbb{P}(\text{target} \rightarrow x | \tilde{\mathcal{F}}_S^{(k-1)} \cap \mathcal{F}_{\sigma_k})$. To expand it further, we need to distinguish whether the destination x is in the set of destinations compatible with the pattern σ_k or not. Evidently, if $x \notin \mathcal{D}_{\sigma_k}$, this term is equal to 0. On the other hand, if $x \in \mathcal{D}_{\sigma_k}$, this term can be computed by simply conditioning the probability distribution $P_S^{(k-1)}(x)$ to the subset of the destinations $\mathcal{D}_{\sigma_k}$ which are compatible with σ_k. We thus obtain the following expression:

$$\mathbb{P}(\text{target} \rightarrow x | \tilde{\mathcal{F}}_S^{(k-1)} \cap \mathcal{F}_{\sigma_k}) = \begin{cases} \frac{P_S^{(k-1)}(x)}{\sum_{y \in \mathcal{D}_{\sigma_k}} P_S^{(k-1)}(y)} & \text{if } x \in \mathcal{D}_{\sigma_k} \\ 0 & \text{if } x \notin \mathcal{D}_{\sigma_k} \end{cases} \tag{6}$$

If we substitute Equations 5 and 6 in 4, we obtain the following recursive structure for the computation of $P_S^{(k)}(x)$:

$$P_S^{(k)}(x) = \begin{cases} \frac{P_S^{(k-1)}(x) \cdot (1 - \gamma_{\sigma_k})}{1 - P_{S*}^{(k-1)}} & \text{if } x \in \mathcal{D}_{\sigma_k} \\ \frac{P_S^{(k-1)}(x)}{1 - P_{S*}^{(k-1)}} & \text{if } x \notin \mathcal{D}_{\sigma_k} \end{cases} \tag{7}$$

A recursive structure for the computation of $P^{(k)}(S)$ can now be obtained as follows:

$$P^{(k)}(S) = \mathbb{P}(\mathcal{F}_S^{(k-1)}) + \mathbb{P}(\mathcal{F}_{\sigma_k} | \tilde{\mathcal{F}}_S^{(k-1)}) \cdot \mathbb{P}(\tilde{\mathcal{F}}_S^{(k-1)}) \tag{8}$$

or, in more compact notation,

$$P^{(k)}(S) = P^{(k-1)}(S) + P_{S*}^{(k-1)} \cdot (1 - P^{(k-1)}(S)) \tag{9}$$

with $P^{(0)}(S) = 0$.

By coupling the equations 5, 7 and 9, we obtain an exact recursive formula for the computation of $P(S) = P^{(\bar{k})}(S)$.

Finally, the approximated expected time to find the target while executing plan S can be computed iteratively as:

$$\begin{aligned} &T^{(k+1)}(S) = T^{(k)}(S) + t_{\sigma_{k+1}} (P^{(k+1)}(S) - P^{(k)}(S)) \\ &T^{(0)}(S) = 0 \\ &T(S) = T^{(\bar{k})}(S) \end{aligned} \tag{10}$$

6 PDDL Model and Planning Mechanism

We express the formal framework described in Section 5 in PDDL2.2 (Edelkamp and Hoffmann 2004) and make use of a high-performing planner, POPF-TIF (Piacentini et al. 2015), coupled with an external solver, to solve the search problem. We describe the model first, then the problems and finally the planning mechanism in the rest of this section.

6.1 PDDL Model

The iterative structure of equations 7, 9 and 10 is reflected in the PDDL model, which contains functions corresponding to the destination probabilities $P_S^{(k)}(x)$, the total probability $P^{(k)}(S)$ and the approximated expected time $T^{(k)}(S)$, both at the current step and at the previous one:

```
(prob ?d - destination)
(previous-prob ?d - destination)
(total-prob)
(previous-total-prob)
(expected-time)
(previous-expected-time)
```

The model includes two types of actions: a flight action that allows the UAV to fly from one waypoint to another and the search actions that correspond to the flight patterns. The actions are durative and their duration is fixed in the problem instance to be the correct (computed) value for the execution of the corresponding search. Each search pattern σ can only be executed when it is active, i.e. during the window of opportunity $[t_\sigma^-, t_\sigma^+]$ that coincides with the period in which the target could plausibly be in the area that the pattern covers.

The effect of a search pattern action, other than to move the UAV from the pattern entry point to its exit point, is to update the destination probabilities $P_S^{(k)}(x)$, the total probability $P^{(k)}(S)$ and the approximated expected time $T^{(k)}(S)$ according to equations 7, 9 and 10. Since the plan is abandoned when the target is discovered, the effect of a search action takes place only when that pattern has failed to discover the target. The planner exploits this information when it adds the next pattern to the plan by decreasing the probability of the destination compatible with the last pattern and increasing the probability of the others. The numerical value of each update is calculated by the external solver, as explain in Section 6. As an example, the following is the description of the action `doSpiral` in PDDL2.2:

```
(:durative-action doSpiral
  :parameters (?from ?to - waypoint ?p - pattern)
  :duration (=?duration (timefor ?p))
  :condition (and
    (at start (beginAt ?from ?p))
    (at start (endAt ?to ?p))
    (at start (at ?from))
    (at start (active ?p)))
  :effect (and
    (at end (at ?to))
    (at start (not (at ?from)))
    (at end (assign (is-last ?p) 1))
    (forall (?d - destination)
        (and (at end (assign (previous-prob ?d) (prob ?d)))))
    (at end (assign (previous-total-prob) (total-prob)))
    (at end (assign (previous-expected-time) (expected-time)))
    (forall (?d - destination)
        (and (at end (increase (prob ?d) (heuristic-change)))))
    (at end (increase (total-prob) (heuristic-approximation)))
    (at end (increase (expected-time) (heuristic-approximation)))))
```

6.2 Problem Instances

The initial state of a planning problem contains all the destinations and the candidate patterns from which the planner chooses the ones to execute. For each destination, the probability at step 0 is specified ($\frac{1}{d}$). Each candidate search pattern is assigned the following information: (i) a fixed duration, which is the computed value for the execution of the corresponding search; (ii) an opportunity window, which specifies when the pattern is active; and (iii) entry and exit waypoints. All these pieces of information are computed during the candidate generation phase based on the geometry of the pattern, the map of the environment and the motion models of the target and the UAV. They are then compiled into a planning problem automatically and fed into the planner together with the domain model.

We use a plan metric that measures the value of the plan in terms of the probability of finding the target and the expected time to do that. The objective function $G(S) = P(S) - kT(S)$ is expressed as follows:

```
(:goal (and (> (total-prob) 0)))
(:metric maximize ( - (total-prob) (* k (expected-time))))
```

The parameter $k \geq 0$ decides the trade-off between probability of discovery and time to complete the mission.

6.3 Planning Mechanism and External Solver

The plans are built using the planner POPF-TIF (Piacentini et al. 2015), built on top of the partial order temporal planner POPF2 (Coles et al. 2010). POPF-TIF handles numeric semantic attachments through the coupling of an external solver. To achieve communication between the planner and the external solver, two groups of numeric state variables are identified, V^{dep} and $V^{special}$. The V^{dep} variables are set by the state progression of the core planner and their values are used by the external solver to calculate the values of the $V^{special}$ variables. These values are in turn used in the next state progression of the planner, while a user-defined linear approximation of the $V^{special}$ variables is employed in the heuristic evaluations.

In our domain, the $V^{special}$ variables calculated by the external solver are the destination probabilities $P_S^{(k)}(x)$, the total probability $P^{(k)}(S)$ and the approximated expected time $T^{(k)}(S)$, respectively indicated as `(prob ?d - destination)`, `(total-prob)`, `(expected-time)`. These quantities depend on their own values in the preceding state and on the last pattern added to the plan. The numeric fluents that keep track of these values represent the V^{dep}: `(previous-prob ?d - destination)`, `(previous-total-prob)`, `(previous-expected-time)` and `(is-last ?p)`. The functions `(heuristic-approximation)` and `(heuristic-change)` are used by the heuristic.

The parameters that are state independent, which are γ_{σ_k} and $\mathcal{D}_{\sigma_k}$ in our case, are given to the external solver in an input file when calling the planner. Our algorithm generates these parameters after the candidate generation phase. In particular, each candidate pattern is assigned a set of compatible destinations, which are selected based on the topology of the map, and a probability of detection γ, which is computed based on several parameters such as the quality of the sensors, the type of terrain that the pattern covers, the distance from the target LKP and the estimated velocity of

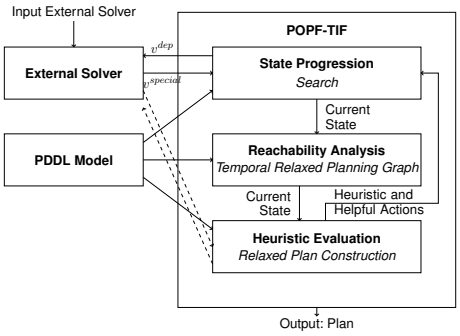

Figure 2: Coupling POPF-TIF with an external solver

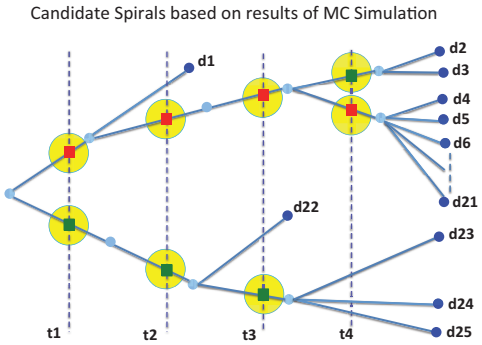

Figure 3: For each time slice, the candidate spirals (in yellow) are generated around the first and second most probable cells.

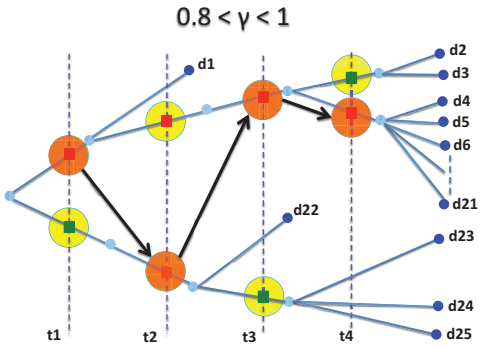

Figure 4: Plan obtained with a high detection probability γ.

the target. Figure 2 shows the planner POPF-TIF and the external solver working in tandem.

We use the anytime version of POPF-TIF which performs cost-improving search: it finds a first solution very quickly, since a plan with one search action is already a feasible solution, but it then spends the additional time improving on this solution by adding further manoeuvres to the plan or by trying different collections of manoeuvres. The plans produced are monotonically improving, so the final plan is selected for execution. We use a time-bounded search limited to 1 minute and a parameter $k = 0.1$ for the objective function $G(S)$, although there are configurable values.

Example: In Figures 3-6, we show pictorially how our hybrid algorithm works. Starting from the prior distribution provided by the MCS, it reasons about the target motion, the results of unsuccessful searches and the effectiveness of search sensors and produces a posterior distribution that flexibly leads the UAV towards promising areas of the map that are still unexplored. The figures highlight how the plan formulated by the planner changes depending on the value of γ (which we assume constant on all patterns to simplify the example). Figure 3 shows the tree $\mathcal{T}$ that connects the LKP node to the destination nodes through their respective shortest paths extracted from the RN. Per each time slice t_i, our algorithm generates two candidate search patterns centred around the nodes that have collected the highest number of particles via MCS (red squares are the nodes with the highest probability, which lay on the top branch that leads to 21 destinations out of 25, and green squares are the second best nodes). Figure 4 shows the plan that is formulated when the detection probability γ is between 0.8 and 1. Since this probability is high, the planner trusts the UAV's sensors and, after failing to find the target in an execution of the first spiral, concludes that the target is not there. Hence, at the next step, it changes strategy by going to explore the other branch of the tree. The same happens at the third step. On the other hand, if the detection probability is lower, between 0.55 and 0.8, before changing strategy, the planner keeps exploring the most promising branch, the top one, and tries the bottom one only at the third step. This is shown in Figure 5. Finally, if the detection probability is poor, below 0.55, the planner ignores the feedback of the sensors and insists on the branch that includes the nodes with the highest probability of discovery (see Figure 6). At first sight, this last plan might seem the best since it contains all the most probable nodes according to MCS. However, let us consider a high detection probability, $\gamma = 0.9$, and calculate the error probability of this plan, which is defined as $E(S) = 1 - P(S)$. Such a probability comes to 0.164. Even assuming perfect detection ($\gamma = 1$), the error probability is still 0.160. On the other hand, the plan shown in Figure 4 has an error probability of 0.002 with $\gamma = 0.9$, which is much lower than both the probabilities obtained for the plan in Figure 6.

7 SaT Simulation

In order to validate our hybrid approach to SaT, we implemented a simulation featuring a fixed-wing UAV involved in a complete SaT mission, building on a similar simulation by Bernardini et al. (2013). The area of operations is a part of Scotland about 100 kilometres square and the target follows a path acquired via Google Maps, with a selected origin and destination (both configurable). The UAV is equipped with an imaging system that allows the target to be observed and that is susceptible to error. Although the exact value of γ that we use in the algorithm depends on the specific pattern, we usually get values around 0.3 for urban terrain, 0.5 for a

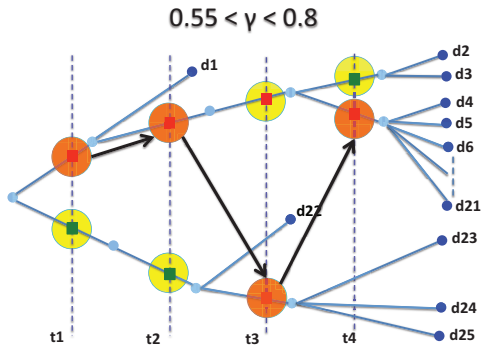

Figure 5: Plan obtained with a good detection probability γ.

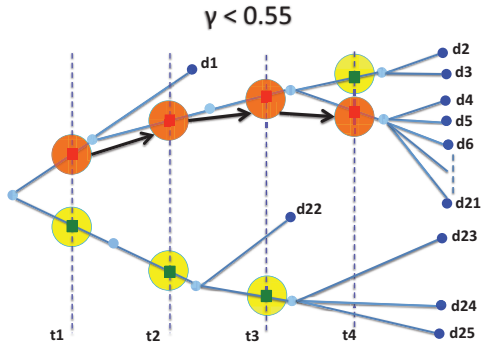

Figure 6: Plan obtained with a poor detection probability γ.

suburban terrain and 0.7 for rough terrain.

During tracking, the UAV simulator follows the target by spiralling over it. Once the target has been lost, the simulator performs the following steps: 1) It runs MCS, which produces a PD map for the target location at different time steps (the most probable cells at the different time check points are visualised in different colours, see Figure 7); 2) It generates a set of candidate search patterns based on the PD map and synthesises a planning task specification featuring these candidates (see yellow circles in Figure 7); 3) It feeds the domain and the problem into the planner and the parameters into the external solver; and 4) After one minute, it dispatches the generated plan to the UAV and simulates its execution (see red circles in Figure 8).

As for MCS, after experimenting with different granularities, we now adopt a grid square size of 500 meters. The graphs extracted from the RN have around 300,000 nodes and 180,000 edges, on average. The target starting point is Stirling. We choose as the set of possible destinations the first 15 most populated cities in Scotland and assign them equal probability. We generate 10,000 particles and, since the total mission time for our application is about one hour and a half, we consider 17 time check points spaced 300 seconds apart. To make the problem more interesting, we assume that, when losing the target, the UAV is blind for 15 minutes. This way the target has time to escape from the area where it has been lost and the UAV needs to enact a smart strategy to rediscover it.

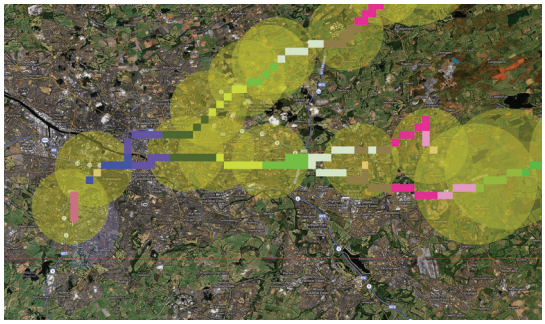

Figure 7: Initial state: circles are search patterns that the planner will consider, which are centred on the grid cells that carry the highest probability of rediscovering the target.

Figure 8: Plan: search patterns selected by the planner for execution.

8 Results and Discussions

We conducted a series of experiments to assess the performance of our hybrid approach to SaT. Currently, it is difficult to compare our approach with RBE methods, as the assumptions behind them are very different from ours (small geographical areas, around 1 km square, and short temporal horizons, around 1-2 minutes). The complexity of RBE methods typically explodes when large instances are considered. Instead, we assess our strategy against comparable ones, which we extract from the work by Bernardini et al. (2013) and Bernardini, Fox, and Long (2015).

In (Bernardini et al. 2013), it has been demonstrated that a plan-based approach outperforms static strategies that involve performing a fixed set of patterns and that are used in several real-world applications (e.g. by BAE Systems Inc.).

Here, we are interesting in showing how the incorporation of Bayesian reasoning into the plan-based SaT approach allows the planner to find better solutions for more complex problems. To this aim, we compare our *hybrid* approach with a plan-based strategy that also uses MCS to produce a target PD map at the time of loss, but that does not update such a map over time, which makes this method oblivious to the history of past searches. We refer to this strategy as *MCS-*

based. After identifying the most probable cells for each time check point, this algorithm generates patterns around the best cells in each time slice. The planner is then in charge of choosing which one to execute and for how many times and their ordering. This method is equivalent to our hybrid approach when the probability of detection is very low. In this case, indeed, the UAV never trusts its sensors and always chooses the patterns that are centred on the most probable cells (see Figure 6).

The MCS-based strategy is known to work very well when the test environment tends to support a greedy searching approach, i.e. when the target moves in an area with a few main roads, when the UAV remains blind for only a short period of time after losing the target (a couple minutes) and when its sensors are rather reliable (Bernardini, Fox, and Long 2015). However, as shown by the experiments below, the performance degrades in more complex situations, and in particular, when the road network is complex. In addition, in our experiments, we prevent the observer from spotting the target for 15 minutes following losing it. This allows us to focus on the benefits of the improved search strategy, by removing the cases where the target is luckily reacquired during the flight to the first search pattern. This effect also accounts for situations in which a target is lost due to entering a no-fly zone area or by exploiting evasive actions. Our hybrid approach is capable of tackling these realistic circumstances by generating plans with high chances of success.

To compare the hybrid and the MCS-based strategies, we generated 15 routes that lead from Stirling (the target starting point) to the fifteen most populated cities of Scotland (Glasgow, Edinburgh, Dundee, East Kilbride, Livingston, Hamilton, Cumbernauld, Dunfermline, Kirkcaldy, Perth, Coatbridge, Airdrie, Falkirk, Motherwell and Rutherglen) and executed the simulation on each route 1000 times (the simulation has a non-deterministic spotting model and target behaviour) for each of the 2 strategies.

Figure 9 shows the proportion of runs in which the target was tracked to its destination against those in which it was ever lost (we remove Perth since the target is never lost on this road). The MCS-based policy has a very low overall success rate, only 10%. On the other hand, the hybrid technique bears an overall success rate of 56%. We achieved an enormous improvement over the MCS approach, despite the difficulty of these problems. The target is usually lost in urban areas where there are many crossroads through which it can take any direction and it has 15 minutes to escape from the UAV vision field. When start searching, the UAV has only a set of hypotheses about the target destination.

Figure 10 displays the average time that the observer tracks the target plotted against journey duration. The hybrid approach tracks the target more successfully, showing that its success in discovering the target is much better than the MCS approach.

Figure 11, which displays how the probability of finding the target changes over time for one specific destination (Edinburgh), demonstrates that the hybrid policy dominates the other and is very robust by offering significant probability of discovery even after more than one hour.

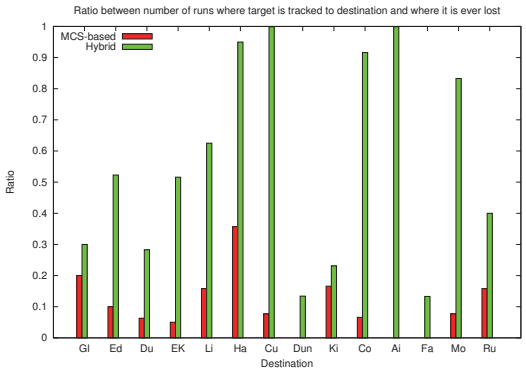

Figure 9: Proportion of the runs in which the target was tracked to its destination against those in which it was lost.

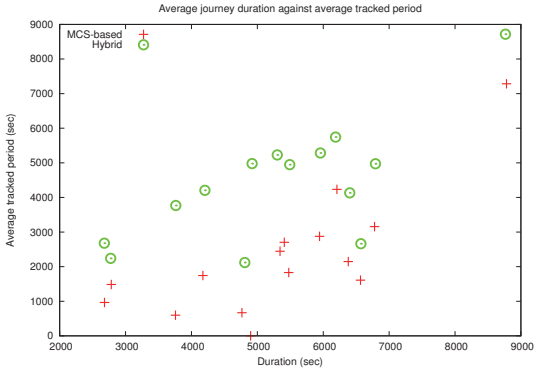

Figure 10: Average time over which the target was tracked to destination against the average journey length.

9 Conclusions and Future Work

We have presented a hybrid approach to autonomous SaT that combines the benefits of using automated planning and probabilistic reasoning. Our experiments show that our strategy is effective in tackling complex real-world SaT missions. Our approach to SaT can be easily modified to support other applications. It can be employed to tackle problems under uncertainty in which MC methods can help in formulating a set of initial hypotheses and planning can be leveraged to prove or disprove the validity of such hypotheses by iteratively updating their probabilities in light of new observations in a Bayesian fashion.

References

Benton, J.; Coles, A.; and Coles, A. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS-12)*.

Bernardini, S.; Fox, M.; Long, D.; and Bookless, J. 2013. Autonomous Search and Tracking via Temporal Planning. In *Proceedings of the 23st International Conference on Automated Planning and Scheduling (ICAPS-13)*.

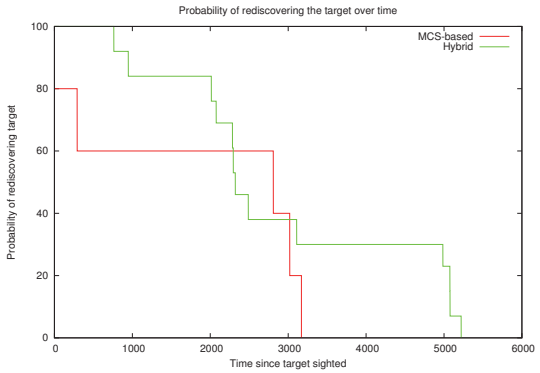

Figure 11: The probability (percentage) of recapturing the target over time while it is heading towards Edinburgh.

Bernardini, S.; Fox, M.; and Long, D. 2015. Combining Temporal Planning with Probabilistic Reasoning for Autonomous Surveillance Missions. *Autonomous Robots*. In Press.

Bourgault, F.; Furukawa, T.; and Durrant-Whyte, H. F. 2006. Optimal Search for a Lost Target in a Bayesian World. In *Field and Service Robotics*, volume 24 of *Springer Tracts in Advanced Robotics*. Springer Berlin. 209–222.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS-10)*.

Edelkamp, S., and Hoffmann, J. 2004. PDDL2.2: The language for the classical part of the 4th international planning competition. In *Proceedings of the 4th International Planning Competition (IPC-04)*.

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

IMO. 2013. *International Aeronautical and Maritime Search and Rescue Manual (IAMSAR)*. United States Federal Agencies.

Lavis, B., and Furukawa, T. 2008. HyPE: Hybrid Particle-Element Approach for Recursive Bayesian Searching and Tracking. In *Proceedings of the 2008 Robotics: Science and Systems Conference*, 135–142.

Lin, L., and Goodrich, M. A. 2014. Hierarchical Heuristic Search Using a Gaussian Mixture Model for UAV Coverage Planning. *IEEE Transactions on Cybernetics* 44:2532–2544.

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

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