Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

Deterministic versus Probabilistic Methods for Searching for an Evasive Target

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
Department of Informatics
King's College London
London, UK, WC2R 2LS
firstname.lastname@kcl.ac.uk

Chiara Piacentini
Department of Mechanical
and Industrial Engineering
University of Toronto
Toronto, Canada, ON M5S
chiara.piacentini@utoronto.ca

Abstract

Several advanced applications of autonomous aerial vehicles in civilian and military contexts involve a searching agent with imperfect sensors that seeks to locate a mobile target in a given region. Effectively managing uncertainty is key to solving the related search problem, which is why all methods devised so far hinge on a probabilistic formulation of the problem and solve it through branch-and-bound algorithms, Bayesian filtering or POMDP solvers. In this paper, we consider a class of hard search tasks involving a target that exhibits an intentional evasive behaviour and moves over a large geographical area, i.e., a target that is particularly difficult to track down and uncertain to locate. We show that, even for such a complex problem, it is advantageous to compile its probabilistic structure into a deterministic model and use standard deterministic solvers to find solutions. In particular, we formulate the search problem for our uncooperative target both as a deterministic automated planning task and as a constraint programming task and show that in both cases our solution outperforms POMDPs methods.

1 Introduction

In a search mission, an aerial vehicle, the observer, wishes to locate the position on the ground of an object, the target. The search problem has been studied extensively for stationary targets or targets moving according to predicable external forces, such as currents and wind, both in the context of operations research (Stone 1975; Benkoski, Monticino, and Weisinger 1991) and that of robotics (He, Bachrach, and Roy 2010; Furukawa et al. 2012; Ragi and Chong 2013; Lin and Goodrich 2014). However, the problem of searching for a target that moves according to its own intentions remains much less explored. We consider here the particularly hard task of searching for a moving target that is *uncooperative* with the observer. We assume that the target moves over a *large* geographical area that offers several opportunities for hiding, which will make it even more difficult for the observer to discover the target.

In this paper, we offer not only an efficient solution for large-scale search for evasive targets, but also a comparative analysis between two completely different approaches to this problem: one exploiting *deterministic* solvers and the other using *probabilistic* reasoning tools. Probabilistic reasoning, and the Bayesian framework in particular, is traditionally used to guide the search process. Based on an initial set of assumptions regarding the target's behaviour, the observer starts exploring the area of operation (AoO). At regular intervals, as it gathers imperfect observations, it updates its knowledge of the object's likely location. As time passes, it explores a larger portion of the AoO and its understanding of the target's position improves, but its resources decrease. Optimising this trade-off is the goal of automated decision making for search. Since search naturally lends itself to this Bayesian formulation, several solutions within this framework have been attempted (Chung and Furukawa 2006; Furukawa, Durrant-Whyte, and Lavis 2007; Lavis and Furukawa 2008). However, we have recently shown that a Bayesian formulation of search can be compiled into a deterministic planning problem and solved with automated planning tools (Bernardini et al. 2016). If a few comparisons between different probabilistic approaches to search have been already carried out (Chung and Burdick 2012), we are not aware of any comparative analysis on alternative paradigms, such as the deterministic and the probabilistic ones.

To carry out this analysis, we adopt the following method (Bernardini, Fox, and Long 2014; 2015). We discretise the search problem in space and in time: we consider a grid laid over the AoO and a set of equally spaced time points from the start to the end of the mission. We make the observer use standard *search patterns* (spirals and lawnmowers) to survey the grid cells in the AoO. If, while executing patterns, the observer finds the target, it abandons its search strategy and starts tracking the target. In this context, the search problem is reformulated as finding a sequence of patterns to execute, $S = (\sigma_1, \ldots, \sigma_{\bar{k}})$, that maximises the probability of finding the target. Extending our previous work (Bernardini et al. 2016), we carefully tailor a probabilistic model to capture this problem. We then encode it in three different ways: (i) as a POMDP; (ii) as an automated planning task; and (iii) as a constraint programming (CP) task. While POMDPs are generally regarded as the elective method for problems under uncertainty, planning and CP are powerful paradigms for deterministic reasoning. When comparing the results, our conclusions are twofold. On the one hand, we show that POMDP solvers, considered to be inefficient on large and complex problems, can in fact deal with them when under-

pinned by expressive models. On the other hand, we provide evidence that, for our search problem, deterministic techniques systematically outperform POMDP methods.

2 Probabilistic Model of Search

We give here a *dynamic programming* formulation of the search task that naturally lends itself to be reformulated both in deterministic and probabilistic terms. We deal with an *evasive* target (see Figure 1). At the beginning of the mission, we assume that the target in unaware of being tracked by the observer. As time passes and it traverses clear regions, it becomes suspicious of being observed and finally starts evading the observer. Although the exact behaviour of the target depends on its destination, it never changes such a destination or stops during its journey. The observer does not know whether the target is trying to evade or not and uses a motion model for it that is described below.

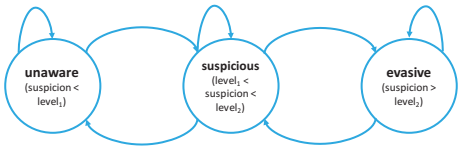

Figure 1: Target's behaviour

2.1 Graph Construction

We assume that the target is located in Euclidean 2-space and that this space is characterised 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 a 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 characterised 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 optimal 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 intersect at least a line 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 line 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: $\mu : \mathcal{D} \to [0, 1]$.

2.2 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 $\Gamma_x = \{\gamma_1, \ldots, \gamma_{(k*l)}\}$ the set of the $(k * l)$ 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 degree 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 $\mathcal{C}(w)$ the set of pairs (x, γ) where $x \in \mathcal{D}$ and γ is a path in $\mathcal{G}_w$ from w to x. We say that these pairs (x, 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: (i) the final destination cell $x \in \mathcal{D}$ is sampled according to the PD μ; (ii) the path $\gamma \in \Gamma_x$ from v_0 to x is sampled according to the PD θ; (iii) $X(t)$ moves with a velocity scaled by a constant factor ω uniformly sampled in the interval [0,1]; (iv) $X(t)$ 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 (v) the jumping time t_k's are iteratively determined according to the following formula: $t_{k+1} - t_k = \delta/\nu_k$, where $\nu_k = \nu^{\min}_{(v_k,v_{k+1})} + \omega(\nu^{\max}_{(v_k,v_{k+1})} - \nu^{\min}_{(v_k,v_{k+1})})$.

2.3 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 checkpoints. 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 $q^{t_k}_v = |\{j|\chi_j(t_k) = v\}|/M$ for $v \in \mathrm{V}$. From the law of large numbers, we know that $q^{t_k}_v$ approximates, for a sufficiently large M, the true marginal distributions of $X(t_k)$.

2.4 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 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 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 shortest and longest time of arrival for the target to the pattern's centre. For every $\sigma \in \Sigma$, 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 $\mathcal{C}_\sigma = \bigcup_{v \in V_\sigma} \mathcal{C}(v)$ the set of all pairs (x, γ) that are compatible with σ. A plan is a sequence of elements in Σ, $S = (\sigma_1, \sigma_2, \ldots, \sigma_{\bar{k}})$, with $\bar{k}$ being its length.

We call $P(S)$ and $T(S)$, respectively, the probability and an approximation of the expected time of finding the target by executing the plan S and $P^{(k)}(S)$ and $T^{(k)}(S)$ the probability and expected time of finding the target within time step k. The approximated expected time is defined by assuming that if the target is discovered by executing the pattern σ, the rediscovery 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:

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

We consider an objective function of type $G(S) = P(S) - kT(S)$, with $k \geq 0$ a constant parameter. Maximisation of $G(S)$ leads to plans that balance, depending on k, probability of rediscovery and time to complete the mission.

2.5 Calculation of Probabilities

To maximise the objective function $G(S) = P(S) - kT(S)$, we need to compute $P(S)$ and $T(S)$. Take a plan with $\bar{k}$ patterns. We have that: $P(S) = P^{(\bar{k})}(S)$ and $T(S) = T^{(\bar{k})}(S)$. Building on (Bernardini et al. 2016), we give a recursive structure for the computation of $P^{(k)}(S)$ and $T^{(k)}(S)$:

$$\begin{aligned} &P^{(k)}(S) = P^{(k-1)}(S) + P_{S*}^{(k-1)} \cdot (1 - P^{(k-1)}(S)) \\ &P^{(0)}(S) = 0 \end{aligned} \tag{1}$$

$$\begin{aligned} &T^{(k+1)}(S) = T^{(k)}(S) + t_{\sigma_{k+1}} (P^{(k+1)}(S) - P^{(k)}(S)) \\ &T^{(0)}(S) = 0 \end{aligned} \tag{2}$$

In Equation 1, $P_{S*}^{(k-1)}$ 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. It is the product of two terms: (i) the probability that the observer finds the target when it is in view, i.e. the detection probability ϕ_{σ_k}; and (ii) the probability that the target has chosen any of the destinations and paths compatible with σ_k (i.e. a pair in $\mathcal{C}_{\sigma_k}$) computed according to the distribution $P_S^{(k-1)}(x, \gamma)$, which encodes the fact that the target has not been discovered earlier. That is:

$$P_{S*}^{(k-1)} = \sum_{(y,\zeta) \in \mathcal{C}_{\sigma_k}} \phi_{\sigma_k} \cdot P_S^{(k-1)}(y, \zeta) \tag{3}$$

More precisely, $P_S^{(k-1)}(x, \gamma)$ represents the probability that the target is driving towards x by following the path γ at time step k after executing the patterns in S and provided that the searches in patterns $\sigma_1, \ldots, \sigma_k$ have failed. To calculate $P_S^{(k-1)}(x, \gamma)$, we need to distinguish whether the pair (x, γ) is in the set of pairs compatible with the pattern σ_k or not. If $(x, \gamma) \notin \mathcal{C}_{\sigma_k}$, this term is equal to 0. On the other hand, if $(x, \gamma) \in \mathcal{C}_{\sigma_k}$, this term can be computed by conditioning the probability distribution $P_S^{(k-1)}(x, \gamma)$ to the subset of the destinations and paths $\mathcal{C}_{\sigma_k}$ which are compatible with σ_k. Hence, we obtain the recursive structure:

$$P_S^{(k)}(x, \gamma) = \begin{cases} \frac{P_S^{(k-1)}(x,\gamma) \cdot (1 - \phi_{\sigma_k})}{1 - P_{S*}^{(k-1)}} & \text{if } (x, \gamma) \in \mathcal{C}_{\sigma_k} \\ \\ \frac{P_S^{(k-1)}(x,\gamma)}{1 - P_{S*}^{(k-1)}} & \text{if } (x, \gamma) \notin \mathcal{C}_{\sigma_k} \end{cases} \tag{4}$$

3 Modelling Search as a POMDP

The probabilistic model described in Section 2 can be naturally encoded as a POMDP.

State S In our problem, we identify two subsystems, the target and the observer, and so the state space is $S = S_{target} \times S_{observer}$. The state of the target is determined by its destination, the path that it is following to reach it, its velocity and the time. We indicate the set of pairs (destination, path) as $\mathcal{C}$, the discretised space of the possible velocity scaling factor ω as Ω and the set of discretised time-points as $\mathcal{T}$. The target's evasive behaviour is captured by the different paths that lead to each destination, which are calculated by sampling the value of the parameter α. The state of the observer is determined by its position in the the grid $\mathcal{X}$ and the time. The position of the observer can be any cell in the grid $\mathcal{X}$, but we can safely consider only the subset of cells from which at least one path of the target is observable, which we call $\tilde{\mathcal{X}}$. Hence, the POMDP state space S is given by $\mathcal{C} \times \Omega \times \mathcal{T} \times \tilde{\mathcal{X}}$. In this representation, the pairs (x, γ) and the velocity scaling factor w are the quantities carrying the uncertainties because they are not known by the observer. Actions, however, do not modify their values.

Actions A The actions space consists of the possible patterns that the observer can perform. In the PODMP model, we do not restrict the observer to choose among a pool of initial candidate patterns. Instead, every cell in $\tilde{\mathcal{X}}$ can be the origin of a pattern. The action space is then $\tilde{\mathcal{X}} \times \tilde{\Sigma}$, where $\tilde{\Sigma} = \{spiral, lawnmower\}$. Given the current state (x, γ, w, τ, c) and the origin c' of a pattern of type $\tilde{\sigma}$, the application of an action $a_{c',\tilde{\sigma}}$ results in a change of the observer position and an update of the time: $a_{c',\tilde{\sigma}}(x, \gamma, w, \tau, c) = (x, \gamma, w, \tau', c')$ where $\tau' = distance\,(c, c')\,/velocity_{observer} + duration_{\tilde{\sigma}}$.

Transition Function T We assume that the position of the observer and its velocity are not subject to uncertainties. The actions are therefore deterministic.

Observations Z The possible observations Z correspond to having seen the target or not during the execution of a

search pattern: $Z = \{seen, not\text{-}seen\}$. The observation $seen$ occurs when the target position is within the area covered by the search pattern executed by the observer and the imaging system is accurate and spots the target. The observation $not\text{-}seen$ occurs in every other case.

Probability of Observation O The conditional probability $O(s, a, z)$ of observing the target while executing a pattern σ depends on the position of σ and the detection probability ϕ_σ.

Reward Function R If the observer makes a observation *seen*, we update the reward by a constant factor. So in our case, $R(s, a) = R(z)$. Although the time when the observation $seen$ is done is not explicitly part of the reward function, POMDP solvers usually evaluate policies based on a discounted reward function and therefore they balance high rewards with time to obtain them.

Belief State $\mathcal{B}$ Since we assume that the observer state is fully observable, its belief states corresponds to the actual states in $S_{observer}$. Instead, the state of the target is unknown and we need to assign an initial PD over the possible target states. We use μ as the PD for the target moving to a destination $x \in \mathcal{D}$, θ as the PD for the target choosing the path $\gamma \in \Gamma_x$ from v_0 to x and a uniforme PD for the velocity factor ω (see Section 2.2).

POMDP Planner We use DESPOT (Somani et al. 2013), a state-of-the-art POMDP planner. It requires a generative model and relies on MCS to update the belief state. Although DESPOT is an online planner, we could use it in an offline mode thanks to the structure of our problem. By forcing all the observations to have the value *non-seen*, we obtain a plan that covers the entire length of the mission.

4 Deterministic Models for Search

The probabilistic model described in Section 2 lends itself to be reformulated as a deterministic model thanks to the specific characteristics of our problem. In search, the uncertainty arises from the unknown position of the target. However, we do not need to model the target position explicitly. When we choose a sequence of patterns to look for the target, we always work under the hypothesis that the target remains undiscovered after executing each pattern. In fact, if the target is found, the observer abandons the plan and switches to tracking. We propose here two different deterministic encodings for search.

4.1 Modelling Search as a Planning Task

In Bernardini et al. (2016), we propose a solution to the search problem for a cooperative target based on automated planning. Building on this work, we formulate search for an evasive target as a deterministic planning task. Based on the target's LKP, the set of destinations $\mathcal{D}$ and the path $\Gamma_{x_1}, \ldots, \Gamma_{x_d}$, the planner populates the plan S by making predictions about which path the target is following and which destination it is aiming for. The planner updates these predictions over time in view of the target's motion and the results of the previous searches. In a Bayesian fashion, at every iteration k, the planner calculates the probabilities $P_S^{(k)}(x, \gamma)$, the total probability $P^{(k)}(S)$ and the approximated expected time $T^{(k)}(S)$ according to equations 4, 1 and 2. Its goal is to produce plans that maximise the objective function $G(S)$.

The probabilistic model described in Section 2 is compiled into a deterministic PDDL (Fox and Long 2003) domain. It contains a fly action, which allows the observer to fly from one location to another, and search actions, which correspond to executing patterns. To keep track of probabilities and expected time, the domain contains functions representing $P_S^{(k)}(x, \gamma)$, $P^{(k)}(S)$ and $T^{(k)}(S)$ both at the current step and at the previous one: `(prob ?d - dest)`, `(previous-prob ?d - dest)`, `(total-prob)`, `(previous-total-prob)`, `(expected-time)` and `(previous-expected-time)`. These functions are updated according to Equations 4, 1 and 2 by the effects of the actions that represent search patterns. The effect of a search action takes place only when that pattern has failed to rediscover the target, since the plan is abandoned when the target is found. Therefore, the planner can exploit this information when it adds the next pattern to the plan by decreasing the probability of the destinations and paths compatible with the last pattern and increasing the probability of the others.

The initial state of a planning problem contains all the destinations and the candidate patterns from which the planner chooses the ones to execute. The goal is empty and the plan metric corresponds to the objective function $G(S)$.

We use the planner POPF-TIF (Piacentini et al. 2015) to build plans. It uses a cost-improving search, so it finds a first solution very quickly, but then improves on it until the time bound is reached. Since our PDDL model involves nonlinear mathematical calculations, which go beyond the scope of modern planners, we couple POPF-TIF with an external advisor to calculate Equations 4, 1 and 2 at each iteration. The probabilities and expected time $P_S^{(k)}(x, \gamma)$, $P^{(k)}(S)$ and $T^{(k)}(S)$ are the variables calculated by the advisor. The parameters that are state independent, which are ϕ_{σ_k} and $\mathcal{C}_{\sigma_k}$, are given to the external advisor in an input file.

4.2 Modelling Search as a CP Task

We formulate the problem of finding a sequence of patterns S among a set of candidates Σ that maximises the probability of discovering the target as a CP task. Due to the complex update of the total probability, the model cannot be linearised and expressed as a mix linear integer programming problem. The model requires the discretisation of the time into a set of τ time-points $\mathcal{T} = \{t_0, ..., t_\tau\}$, where $t_\tau \geq \max_{\sigma \in \Sigma} t_\sigma^+$. A binary variable $z_{\sigma,t} = \{0, 1\}$ indicates if a pattern σ is being executed at time-point t. The CP problem is to find the values of $z_{\sigma,t}$, $\forall\ \sigma \in \Sigma$, $\forall t \in \mathcal{T}$ that maximise the objective function $G(S) = P(S) - kT(S)$. A set of constraints must be imposed to make sure that the sequence of patterns can be executed by the observer, as shown in Model 1. Constraint (M1.1) ensures that only one pattern at a time can be executed, while constraint (M1.2) indicates that a pattern can be executed only within its predefined time

window. In addition, two patterns σ and λ can be performed one after the other only if the sum of the time needed to execute σ and the time needed to reach λ has elapsed, as shown by constraint (M1.3). Constraints (M1.4-7) update the total probability and the expected time needed to calculate the objective function, while constraints (M1.8-12) represent the initial state.

Model 1 CP Formulation of the Search Problem.

Set:

Σ (set of candidate patterns), $\mathcal{T}$ (set of time-points), $\mathcal{C}$ (set of pairs (x, γ))

Parameters:

$t^{fly}_{\sigma_i,\sigma_j}$ (time to fly from σ_i to σ_j), t^{exe}_{σ} (duration of the execution of the pattern σ), t^{+}_{σ} (minimum time at which σ can be executed), t^{-}_{σ} (maximum time at which σ is executed), ϕ_σ (detection probability of σ), $\delta_{c,\sigma}$ (coefficient indicating if $c \in \mathcal{C}_\sigma$)

Variables:

$z_{\sigma,t} \in \{0, 1\}$ (1 if σ is executed at time t, 0 otherwise), $P_{(x,\gamma),t} \in [0, 1]$ (prob. at time step t that target is going to destination x via path γ), $P^{\sigma^*}_t \in [0, 1]$ (prob. that the target is found at time t while executing pattern σ^*), $P_t \in [0, 1]$ (total prob. of finding the target at time t), $T^{exp}_t \in [0, \max_\sigma t^{+}_{\sigma}]$ (expected time to find the target at time-step t)

Maximise:

$$P_{t_{|\mathcal{T}|}} + w \cdot T^{exp}_{t_{|\mathcal{T}|}}$$

Subject to:

$$\sum_{\sigma\in\Sigma} z_{\sigma,t} \leq 1 \qquad \forall t \in \mathcal{T} \qquad \text{M1.1}$$

$$z_{\sigma,t} = 0 \qquad \forall \sigma \in \Sigma, \forall t \in \mathcal{T} | t < t^{-}_{\sigma} \wedge t \geq t^{+}_{\sigma} \qquad \text{M1.2}$$

$$z_{\sigma,t} + \sum_{i=t}^{t+t^{exe}_{\sigma}+t^{fly}_{\sigma,\lambda}} z_{\lambda,i} \leq 1 \qquad \forall \sigma, \lambda \in \Sigma, \forall t \in \mathcal{T} \qquad \text{M1.3}$$

$$P^{\sigma*}_t = \sum_{\sigma\in\Sigma} \sum_{c\in\mathcal{C}} \left(P_{c,t-1} \phi_\sigma \delta_{c,\sigma}\right) z_{\sigma,t} \qquad \forall t \in \mathcal{T}/\{t_0\} \qquad \text{M1.4}$$

$$P_{c,t} = P_{c,t-1} \left(1 - \sum_{\sigma\in\Sigma} \left(1 - \frac{1-\phi_\sigma \delta_{c,\sigma}}{1-P^{\sigma*}_t}\right) z_{\sigma,t}\right) \qquad \forall t \in \mathcal{T}/\{t_0\}, c \in \mathcal{C} \qquad \text{M1.5}$$

$$P_t = P_{t-1} + \sum_{\sigma\in\Sigma} P^{\sigma^*}_t \left(1 - P_{t-1}\right) z_{s,t} \qquad \forall t \in \mathcal{T}/\{t_0\} \qquad \text{M1.6}$$

$$T^{exp}_t = T^{exp}_{t-1} + \sum_{\sigma\in\Sigma} \frac{t^{-}_{\sigma}+t^{+}_{\sigma}}{2T} \left(P_t - P_{t-1}\right) z_{\sigma,t} \qquad \forall t \in \mathcal{T}/\{t_0\} \qquad \text{M1.7}$$

$$z_{0,0} = 1 \qquad \text{M1.8}$$

$$\delta_{0,c} = 0 \qquad \forall c \in \mathcal{C} \qquad \text{M1.9}$$

$$P_0 = 0 \qquad \text{M1.10}$$

$$\delta_{c,0} = \frac{1}{|\mathcal{C}|} \qquad \forall c \in \mathcal{C} \qquad \text{M1.11}$$

$$T^{exp}_0 = 0 \qquad \text{M1.12}$$

Model 1 can be solved with any off-the-shelf CP solver. We used IBM ILOG CPLEX CP Optimizer v12.6 (IBM 2013) a software designed to solve constraint satisfaction and optimisation problems. The solver performs an initial constraint propagation that reduces the search space size. The self-adapting large neighbourhood search algorithm is then triggered on the reduced space in combination with constraint propagations until a solution is found (Laborie and Godard 2007; Laborie and Rogerie 2008).

5 Experimental Results

We conducted extensive experiments to assess the performance of our three approaches to search. To do that, we developed a simulator of a fixed-wing UAV in collaboration with our industrial partners, BAE Systems (Bernardini et al. 2013). We abstract away from issues concerning control and stability of flight and assume that the UAV is capable of executing accurately a specified set of manoeuvres. The technique used to generate these manoeuvres is transparent to the rest of the simulation and so we can easily plug in our three different techniques. All solvers are given 1 minute to generate the plans. The AoO is a fragment of Scotland about 100 kilometres square and the target follows a path acquired via GraphHopper[1] by setting up origin and destination. The target starting point is Stirling and the possible destinations are the first 15 most populated cities in Scotland, which are assigned equal probability. The UAV is equipped with an imaging system that allows it to observe the target and it is susceptible to error. Although the exact value of the detection probability depends on the pattern, we usually use values around 0.3 for urban terrain, 0.5 for a suburban terrain and 0.7 for rough terrain. The terrain type influences also the concealment level η, which is higher in forested and urban terrains and lower in suburban and rough terrains. The target starts unaware of being followed and therefore it moves towards its destination according to the quickest path. It starts to evade when the observer is within its line of sight for a prolonged time. When evading, the target chooses a route r towards its destination that maximises the total concealment level, obtained as the sum of the values of η on the different segments of r.

Figure 2-Left shows the ratio of runs in which the target was tracked to its destination against total number of runs. The POMDP policy has an overall success rate of 33.13%, the planning-based technique 49% and the CP-based one 51.13%. The deterministic approach significantly outperforms the PODMP approach in all paths except for one. The planning and CP approaches have similar performance.

Figure 2-Center displays the average time that the observer tracks the target against journey duration and shows that the deterministic approaches track the target much more successfully than the POMDP approach. They spend more time tracking the target because their search is shorter and more effective.

Figure 2-Right displays the probability of finding the target over time for one specific destination (Cumbernauld). It demonstrates that the plan-based and CP policies dominates the POMDP one and are very robust as they offer significant probability of discovery even after more than 15 minutes.

Figure 3 shows search time versus tracking time during the target's journey to Cumbernauld. Comparing these times, it is easy to see that when we use one of the deterministic approaches, the target is usually quickly rediscovered after being lost, which implies that the observer spends most of its time tracking the target. On the other hand, the POMDP approach spends a lot of time in searching the target, often never reacquiring it after the first loss.

Planning vs CP We use a coarser time discretisation for the CP model than for planning to keep the size of the task manageable for the CP solver. This coarser discretisation results in plans with fewer search patterns (CP usually selects around half the number of patterns w.r.t. planning). In consequence, as the mission time is the same for all the approaches, when the observer executes a plan provided by the CP solver, it spends longer on the first few patterns in the plan, which are those that carry a higher probability of rediscovering the target. Although this choice is beneficial in

[1] https://graphhopper.com - Accessed: 2016-09-11

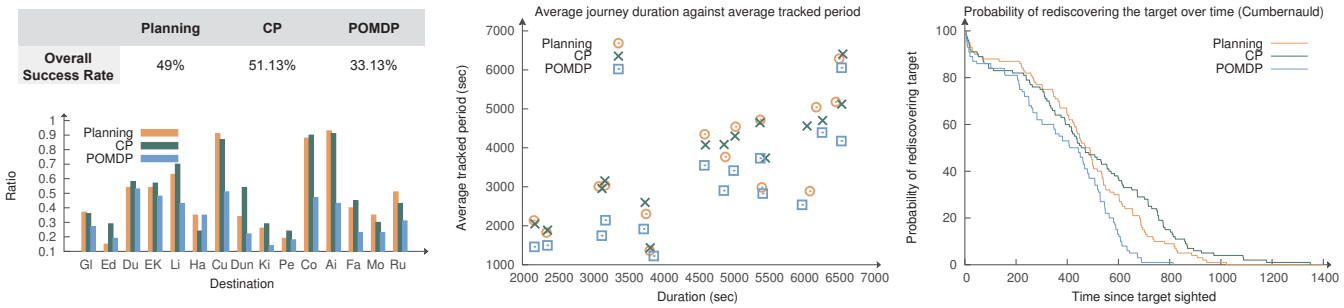

Figure 2: *Left:* Ratio of runs in which target was tracked to destination. *Center*: Average time over which target was tracked to destination against journey length. *Right*: Probability of recapturing target over time while it is going to Cumbernauld.

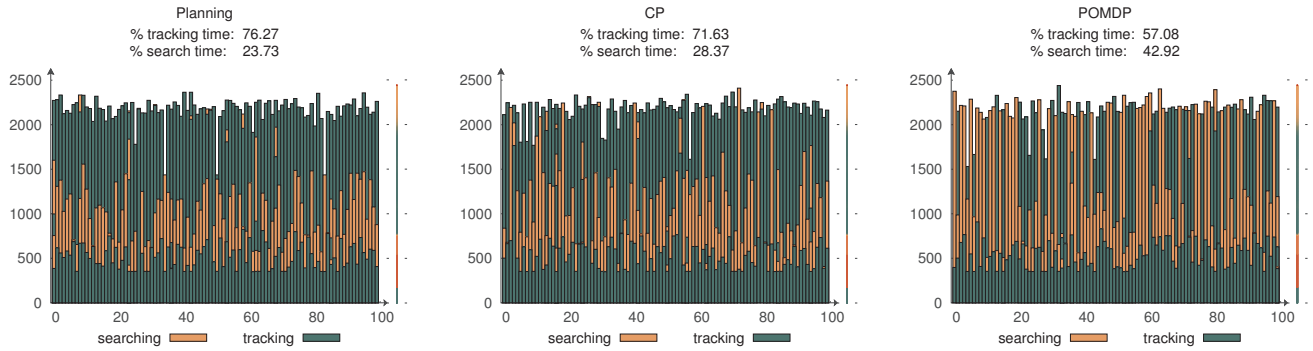

Figure 3: Search time versus tracking time during target's journey to Cumbernauld for 100 runs.

our domain and accounts for the good performance of CP, if a fine discretisation is needed (because for example the domain contains short actions like communication activities), then the planning approach will scale better than the CP one.

POMDP vs deterministic approaches The POMDP and the deterministic models share a number of similar features: the target motion model, the use of MCS, the iterative update of probabilities. Their different use of them, however, accounts for the discrepancy in the overall performance of the two approaches. In planning and CP, the motion model is used to select at the outset a pool of candidate patterns among which to pick those to execute. When the plan is constructed, the motion model is no longer considered explicitly, but underlies the pattern activation time windows. Such windows are used in the calculation of the expected time of finding the target, but not in the update of the total probability. On the other hand, the POMDP model continuously reasons about the exact position of the target to calculate the policy that maximises the reward associated with finding it. Therefore, the POMDP aims to gain a precise understanding of the location of the target, which is reflected in the positioning of the search patterns. This makes the PODMP's performance more sensitive than the deterministic methods to the alignment between the model and the real motion of the target. If the algorithm run time is not an issue and the problem at hand involves a small region and a predictable target, the POMDP method is effective since it manages to build an accurate target motion model, hence placing patterns precisely. However, in the problem we consider, which involves an evasive target moving in a large areas and a tight run time bound of one minute, the POMPD approach suffers the mismatch between modelled and real target motion and its performance deteriorates w.r.t. deterministic methods, which remain more robust to such mismatch.

6 Conclusions

In this paper, we focus on the problem of a searching agent with imperfect sensors that seeks to locate an evasive target in a large geographical area. Effective solutions to this problem are instrumental to fully exploit the potential of autonomous aerial vehicles in several civilian and military missions (e.g. surveillance and search-and-rescue). After discretising the problem in space and time, we propose a dynamic programming characterisation of the search task that captures its intrinsic probabilistic structure. Based on this formulation, we then compare two alternative approaches to find solutions. On the one hand, we compile away uncertainty and use standard deterministic algorithms to solve the problem, particularly planning and CP. On the other hand, we transform the model in a POMDP and use a state-of-the-art POMDP solver against it. We provide evidence that deterministic solutions systematically outperform POMDPs methods, although these are generally regarded as the elective solution for complex problems under uncertainty.

References

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

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)*.

Bernardini, S.; Fox, M.; Long, D.; and 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-16)*.

Bernardini, S.; Fox, M.; and Long, D. 2014. Planning the Behaviour of Low-Cost Quadcopters for Surveillance Missions. In *Proceedings of the 24st International Conference on Automated Planning and Scheduling (ICAPS-14)*.

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

Chung, T. H., and Burdick, J. W. 2012. Analysis of search decision making using probabilistic search strategies. *IEEE Transactions on Robotics* 28(1):132–144.

Chung, C. F., and Furukawa, T. 2006. Coordinated Search-and-Capture Using Particle Filters. In *Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision (ICARCV '06)*, 1–6.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20.

Furukawa, T.; Mak, L. C.; Durrant-Whyte, H. F.; and Madhavan, R. 2012. Autonomous Bayesian Search and Tracking, and its Experimental Validation. *Advanced Robotics* 26(5-6):461–485.

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

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.

IBM. 2013. CP Optimizer User's Manual. 130.

Laborie, P., and Godard, D. 2007. Self-adapting large neighborhood search: Application to single-mode scheduling problems. *Proceedings MISTA-07, Paris* 8.

Laborie, P., and Rogerie, J. 2008. Reasoning with Conditional Time-Intervals. *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference, May 15-17, 2008, Coconut Grove, Florida, USA* 555–560.

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.

Ragi, S., and Chong, E. K. P. 2013. UAV Path Planning in a Dynamic Environment via Partially Observable Markov Decision Process. *IIEEE Transactions on Aerospace and Electronic Systems* 49:2397–2412.

Somani, A.; Ye, N.; Hsu, D.; and Lee, W. 2013. DESPOT : Online POMDP Planning with Regularization. *Advances in Neural Information Processing Systems* 1–9.

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

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