Autonomous Robots (2020) 44:831–843
https://doi.org/10.1007/s10514-019-09900-0

Graph search of a moving ground target by a UAV aided by ground sensors with local information

Krishna Kalyanam[1] · **David Casbeer**[2] · **Meir Pachter**[3]

Received: 2 January 2019 / Accepted: 26 December 2019 / Published online: 9 January 2020

Abstract
The optimal control of a UAV searching for a target moving, with known constant speed, on a road network and heading toward one of many goal locations is considered. To aid the UAV, some roads in the network are instrumented with unattended ground sensors (UGSs) that detect the target's motion and record the time it passes by the UGS. When the UAV flies over an UGS location, this time stamped information, if available, is communicated to it. At time 0, the target enters the road network and selects a path leading to one of the exit nodes. The UAV also arrives at the same entry UGS after some delay and is thus informed about the presence of the target in the network. The UAV has no on-board sensing capability and so, capture entails the UAV and target being colocated at an UGS location. If this happens, the UGS is triggered and this information is instantaneously relayed to the UAV, thereby enabling capture. On the other hand, if the target reaches an exit node without being captured, he is deemed to have escaped. We transform the road network, which is restricted to a directed acyclic graph, into a time tree whose node is a tuple comprising the UGS location and evader arrival time at that location. For a given initial delay, we present a recursive forward search method that computes the minimum capture time UAV pursuit policy, under worst-case target action. The recursion scales poorly in the problem parameters, i.e., number of nodes in the time tree and number of evader paths. We present a novel branch and bound technique and a pre-processing step that is experimentally shown to reduce the computational burden by at least two orders of magnitude. We illustrate the applicability of the proposed pruning methods, which result in no loss in optimality, on a realistic example road network.

Keywords Pursuit evasion · Partial information · Graph search · Brand and bound

1 Introduction

We are concerned with capturing a ground target moving on a road network. The operational scenario is as follows. The access road network to a restricted (protected) zone, patrolled by an Unmanned Air Vehicle (UAV), is instrumented with Unattended Ground Sensors (UGSs); placed at critical locations. The problem at hand is motivated by a base defense/asset protection scenario, where a UAV performs the role of a pursuer aided by the UGSs. The UAV is typically equipped with a gimballed camera, that is used to "capture" visual imagery of the ground target (evader). This imagery is passed on to a human operator, who classifies the target as a potential threat or otherwise and initiates further action. As the ground target, referred to hereafter as the evader, passes by an UGS, the UGS is triggered. A triggered UGS turns, say, from *green* to *red* and records the evader's time of arrival. We assume that there is no observation error, i.e., the UGS accurately reports the time of arrival. We also assume that the UGS has zero false alarm and missed detection rates. We assume that the layout of the road network and the placement of the UGSs is known to the pursuer/UAV. We also assume that the target travels at a constant speed that is known to the pursuer. When the pursuer arrives at an UGS

✉ Krishna Kalyanam
krishnak@ucla.edu

David Casbeer
david.casbeer@us.af.mil

Meir Pachter
meir.pachter@afit.edu

[1] Palo Alto Research Center (PARC), a Xerox Company, Palo Alto, CA 94304, USA

[2] Control Science Center of Excellence, Air Force Research Laboratory (AFRL/RQQA), Wright-Patterson A.F.B., Dayton, OH 45433, USA

[3] Department of Electrical and Computer Engineering, Air Force Institute of Technology (AFIT/ENG), Wright-Patterson A.F.B., Dayton, OH 45433, USA

location, the information stored by the UGS is uploaded to the pursuer, namely, the green/red status of the UGS and, if the UGS is red, the evader's arrival time at the UGS. The evader can be captured in one of two ways: either the evader and pursuer synchronously arrive at an UGS location, or the pursuer is already loitering above an UGS location when the evader arrives there. In both cases, the UGS is triggered, and this information is instantaneously relayed to the pursuer enabling capture on camera. The evader's goal is to reach certain restricted locations on the road network without being captured on camera by the UAV. Due to the UAV's information pattern, which is restricted to partial observations of the physical state of the dynamic game, we are running into the difficulties brought about by the *dual control* effect (Başar 2001), where the current information state determines the UAV's optimal control while at the same time the information that will become available to the UAV will be in part determined by his current control.

1.1 Literature review

We are dealing with a pursuit evasion problem, with partial information, on a graph. This is a variant of the classic discrete pursuit-evasion game introduced by Parsons (1978). Discrete pursuit-evasion games are also known as graph search problems, wherein a pursuer and evader take turns moving from vertex to vertex on a graph. Many variations to this problem have been studied over the years (see Fomin and Thilikos 2008; Chung et al. 2011; Robin and Lacroix 2016 for survey and detailed taxonomy). Algorithms and time complexity results for a variety of graph-based pursuit evasion problems are provided in Borie et al. (2011). Typical formulations of discrete pursuit-evasion games deal with evaders that are either invisible (Dereniowski et al. 2015) or always visible to the pursuer (Chung et al. 2011). In practical scenarios, a more plausible assumption is that the pursuer has limited sensing capability (Kehagius et al. 2014; Clarke et al. 2017). Optimal strategies on a square grid, where the pursuer has line of sight visibility is provided in Sugihara and Suzuki (1989), Dumitrescu et al. (2010). The role of information available to the players, when the pursuer has limited or no visibility of the evader is investigated in Isler and Karnad (2008). Networked robotic systems, where distributed sensors on the ground provide sensing-at-a-distance capability have been considered in Vieira et al. (2009), Sinopoli et al. (2003). However, the information model considered therein is global i.e., the sensor data from all sensors are instantaneously relayed over a global communication network to all pursuers. A robust target tracking algorithm for a mobile pursuer with a fan-shaped field of view and finite sensing range with probabilistic guarantees is provided in Oh et al. (2018).

A related application to our work, where an Unmanned Surface Vehicle (USV) efficiently blocks the advancement of an intruder boat toward a valuable target is considered in Svec and Gupta (2012). There is little record in the literature on the information model we have adopted i.e., delayed position information about the target is available to the UAV locally. This is a natural model that occurs for example in police detective work, where a crime occurs and the police arrive at the scene and have to find and prosecute the criminal. The information obtained from witnesses are delayed and possibly out of sequence making it a difficult estimation problem. Our information model has recently been embraced for a data driven estimation problem considered in Carlone et al. (2018). A related work where witness reports about an evader are immediately made available to the pursuer is considered in Clarke (2009). In the past, a setup where the evader's location information is available to the pursuer after a known constant delay has been considered (Dzyubenko and Pshenichnyi 1972). For the problem at hand, the delay seen by the pursuer is a function of his past actions and therefore, not a constant. In Demirbas et al. (2003), the authors impose the additional constraint that the sensor data is local; however they allow for the sensors close to the evader to communicate with each other and maintain a "tracking" tree that is rooted at the evader. We note that the distributed sensing and local information model sets our work apart from the rest of the literature on pursuit evasion on graphs. Indeed, we are dealing with a deterministic pursuit-evasion game on a graph where the evader's strategy is open-loop control and the pursuer has partial information.

1.2 Prior work

In the past (Krishnamoorthy et al. 2013a, b), we considered a discrete time setting and a highly structured graph i.e., a Manhattan grid, for which analytic solutions were derived for the minimum time (steps) to capture. In the current setting where time is a continuous variable and the evader travels with a known constant speed on a Directed Acyclic Graph (DAG), we provided preliminary results to the minimum time pursuit problem in Krishnamoorthy et al. (2015). Another performance metric of interest in the current setting is the maximal delay after which the UAV can arrive at the entry UGS and guarantee capture of the evader. In Sundaram et al. (2017), we provide complexity results for our problem setting and show that it is NP-hard to determine whether the UAV can enter the network with some nonzero delay and still be guaranteed to capture the evader. Furthermore, we also show that it is NP-hard to approximate (within any constant factor) the largest delay at which the pursuer can enter and still guarantee capture. Regardless, a priori knowledge of the maximal delay at each node in the graph reduces the computational burden for the minimum time pursuit problem. We shall elaborate on this advantage in a later section (see Sect. 3.3). For the case of an evader moving at a known constant speed

Table 1 Summary of past work by the authors on the topic with details on problem specification and solution methods

Reference	Time	Graph	Speed	Metric	Comments
Krishnamoorthy et al. (2013a)	Discrete	Manhattan grid (3 rows)	Pursuer is 2× faster	Minimum steps	Analytical solution
Krishnamoorthy et al. (2013b)	Discrete	Manhattan (infinite) grid	2 Pursuers (2× faster)	Minimum steps	Analytical solution
Krishnamoorthy et al. (2016b)	Continuous	Directed acyclic graph	Constant target speed	Maximal delay	Exact solution
Krishnamoorthy et al. (2016a)	Continuous	Directed acyclic graph	Bounded target speed	Maximal delay	Exact solution
Chen et al. (2017)	Continuous	Directed graph	Constant target speed	Minimum time	Approximate solution
Rasmussen et al. (2016)	Continuous	Real world road network	SUV (ground target)	–	2 UAV autonomy demo
Sundaram et al. (2017)	Continuous	Directed acyclic graph	Constant target speed	Maximal delay	Complexity analysis
Krishnamoorthy et al. (2015)	Continuous	Directed acyclic graph	Constant target speed	Minimum time	Preliminary results
This paper	Continuous	Directed acyclic graph	Constant target speed	Minimum time	Exact solution

on a DAG, we present an optimal *max min* recursion that computes the maximal delay and corresponding pursuit policy (Krishnamoorthy et al. 2016b). In Krishnamoorthy et al. (2016a), we extend this result to the case where the evader's speed can vary arbitrarily within positive bounds known to the UAV. In Chen et al. (2017), we relax the assumption of a directed acyclic graph and provide a finite time horizon based sub-optimal solution. The practical aspects of our problem, with a full blown field demonstration of two UAVs cooperatively pursuing and capturing (on camera) a ground target, are detailed in Rasmussen et al. (2016). Table 1 provides a list of papers that the authors published on this topic with details on the problem setup and solution methods considered in each paper.

This paper extends the preliminary work presented in Krishnamoorthy et al. (2015) and incorporates the following significant improvements:

1) A rigorous mathematical formulation of the underlying optimization problem.
2) A reorganization of the main result via a series of Lemmas characterizing the optimal solution that ultimately lead to the recursive forward search method.
3) A branch and bound technique and maximal delay based preprocessing step that reduces the computational burden by two orders of magnitude.
4) A real world example that clearly demonstrates the applicability of the proposed method.

The remainder of the paper is organized as follows. We set up the model and assumptions in Sect. 2. This is followed by the performance metric of interest and the corresponding optimization problem in Sect. 3. The main focus of the paper i.e., a restriction to the action space that maintains optimality and results in a reduced computational burden is provided in Sect. 3.2. We acknowledge that Sects. 3.1 and 3.2 bear striking similarity to the analytical treatment espoused in our earlier work (Krishnamoorthy et al. 2016b). However, the key difference is the performance metric of interest which results in a completely different algorithm to compute the optimal pursuit policy. Indeed, a novel branch and bound technique that results in a significant reduction in the computational burden is prescribed in Sect. 3.4. In Sect. 4, we validate the solution method on an example problem. The savings in computational burden is illustrated on a real world example problem in Sect. 4.2. Finally, we provide some concluding remarks and directions for future research in Sect. 5.

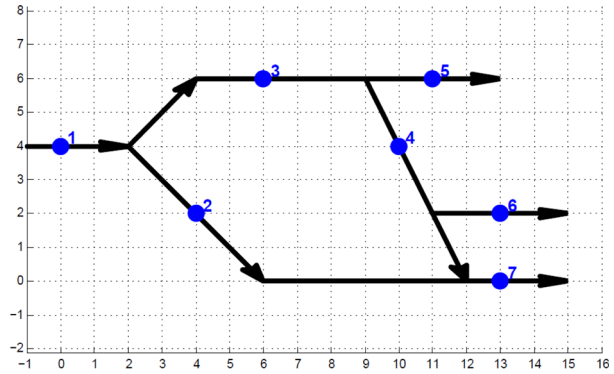
2 Model and assumptions

Figure 1a depicts an illustrative road network with distances prescribed by the embedded coordinate system. The roads along which the evader travels are shown in black (arrows indicate direction of travel) and the numbered UGSs are shown as solid blue circles. In this example, 1 is the entry UGS into the road network and 5, 6 and 7 are the exit UGSs. The environment is modeled as a multi-graph and we assume that the pursuer can travel between any two UGSs.

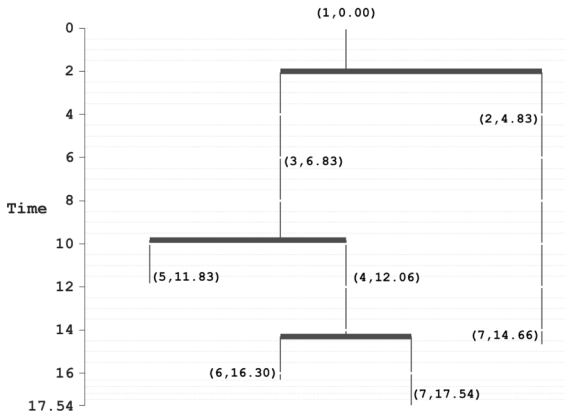
2.1 Evader model

Let the road network be defined by a Directed Acyclic Graph (DAG) consisting of UGSs numbered $1, \ldots, m$. Let the evader enter via UGS 1 at time 0 and travel, without loss of generality, at unit speed towards one of the exit UGSs. Furthermore, let there be n ($\geq$ 1) possible evader paths emanating from the entry UGS 1 and terminating at an exit UGS. A complete list of symbols (and meaning) that appear in this text is provided in Table 2. We use the index, $\rho = 1, \ldots, n$ to indicate a specific path. In the example (see Fig. 1a), we have the evader paths $\rho = 1, \ldots, 4$ given by $1 \rightarrow 3 \rightarrow 5, 1 \rightarrow 3 \rightarrow 4 \rightarrow 6, 1 \rightarrow 3 \rightarrow 4 \rightarrow 7$ and $1 \rightarrow 2 \rightarrow 7$ respectively.

If UGS j can be reached from UGS i by traveling along path ρ, we let $d_e(i, j; \rho)$ indicate the distance between the



(a) Road network on a grid with coordinates



(b) Time tree showing UGS location and time of evader arrival in parenthesis

Fig. 1 Example road network: **a** grid and **b** 4 possible evader paths

two. We simply use $d_e(i, j)$ when the path index is self-evident or if every path between i and j is of the same length. Since the evader travels at unit speed, $d_e(1, j; \rho)$ also indicates the evader's time of arrival at UGS j; if he chooses to travel along path ρ.

For ease of exposition, we unwrap the road network (Fig. 1a) into the tree G shown in Fig. 1b, where time evolves as we go down the tree from the root node to the leaf nodes. Let the set of nodes in the tree be given by $\mathcal{U}$. A node $p \in \mathcal{U}$ is a tuple, i.e., $p = (\ell, a)$, where $\ell = 1, \ldots, m$ indicates an UGS location and a indicates the evader time of arrival at that UGS. Note that the number of nodes in the time tree depends on the number of UGSs and degree of the original road network (DAG). In the example (see Fig. 1b), we have 7 UGSs but $|\mathcal{U}| = 8$ since the evader can reach exit UGS 7 at two different times by taking either path 3 or 4. In this work, we shall assume a finite number of UGSs and furthermore, all evader paths are of finite length. This results in a finite number of nodes in the time tree G. For a node, $u = (\ell(u), a(u)) \in \mathcal{U}$, let:

$$\mathbf{P}(u) = \{\rho : d_e(1, \ell(u); \rho) = a(u), \rho = 1, \ldots, n\}, \tag{1}$$

indicate the set of all paths the evader can take to arrive at UGS $\ell(u)$ at time $a(u)$.

The tacitly assumed information pattern is that the evader has no situational awareness, which is equivalent to saying that the evader decides what path he will take at time 0. The simplifying assumption here is that the evader has prior knowledge of the road network but is unaware of the UGS locations and also oblivious to the presence of the UAV. This critical assumption helps avoid complications arising out of adversarial evader actions and the resulting game scenario. The assumption is also supported by practical considerations (Rasmussen et al. 2016), wherein the UGSs would be typically buried or otherwise hidden from plain sight. Moreover, the pursuer envisioned here is a small (wingspan of 5 ft) electric motor driven UAV flying at a high altitude (500–1000 ft) and hence, is very difficult to sight from the ground. So, in the absence of any reliable feedback information, the evader is forced to operate in open loop. Due to the absence of active evasive actions, we shall revert to calling the evader a (ground) target.

2.2 Pursuer model

The UAV's travel time from UGS i to UGS j is given by a metric, $d_p(i, j)$ that satisfies the triangle inequality:

$$d_p(i, j) \leq d_p(i, h) + d_p(h, j), \tag{2}$$

for any $i, j, h = 1, \ldots, m$ and $d_p(j, j) = 0, \forall j \in \mathcal{U}$. In this way, we allow for the UAV and evader to travel on different edges of the original multi-graph road network. We will further assume that the UAV has an advantage over the target by the following means:

$$d_p(i, j) < d_e(i, j; \rho), \forall i, j, \rho; i \neq j. \tag{3}$$

It is clear that if there is no such advantage, the UAV can never catch up with the target and hence, capture would be impossible. Note that (3) does not necessarily require the UAV to be faster. It could simply be that the flight path between two UGSs is shorter than the road connecting the two.

The UAV is able to make decisions only after it gains new information at a node in the time tree. Let the UAV arrive at UGS 1 at time $t_0 > 0$. At this time, the UAV is only aware that the target has entered the road network and could possibly be on any one of the n paths that emanate from UGS 1. So, the initial path (uncertainty) information available to the UAV is given by $\mathbf{P}((1, 0)) = \{1, \ldots, n\}$—see (1) for definition. Knowing the path information set and current time is equivalent to maintaining the set of all possible target locations (on the road network) given the unit target speed. We realize that a constant target speed assumption may be unrealistic;

Table 2 Nomenclature of symbols in order of appearance

Notation	Meaning
m	Number of UGSs
n	Number of evader paths
ρ	Path index that takes values from $1, \ldots, n$
$d_e(i, j; \rho)$	Target travel time between UGS i and UGS j along path ρ
$u = (\ell(u), a(u))$	Node in time tree indicating UGS location $\ell(u)$ and target time of arrival $a(u)$
$\mathcal{U}$	Set of all nodes in time tree
$\mathbf{P}(u)$	Set of all paths target can take to arrive at UGS location $\ell(u)$ at time $a(u)$
$d_p(i, j)$	UAV travel time between UGS i and UGS j
t_k	Discrete decision epochs where $k = 0, 1, 2, \ldots$
y_k	*Red* or *green* observation made at decision epoch t_k
$\mathcal{P}_k$	Target path information available to UAV at decision epoch t_k
$\ell(p_k)$	UAV location at decision epoch t_k
u_k	UAV control action taken at decision epoch t_k
s_k	Information state at decision epoch t_k
$\delta(s_k, u_k)$	Time difference between decision epochs t_k and t_{k+1}
π	Finite sequence of non-repeating nodes (UAV policy)
κ	Chosen target path (target policy)
$e(\rho)$	Target arrival time at exit node along path ρ
$\mathcal{C}(s)$	Set of capture nodes corresponding to information state s
$\mathcal{M}(s_k)$	Worst case minimum time to capture from information state s_k
$J(\pi, \kappa)$	Performance metric for policies (π, κ)
$Y(s_k, u_k)$	Set of possible observations at UGS $\ell(u_k)$ given the information state s_k
$\mathcal{D}(\ell \vert \mathcal{P})$	Latest time UAV can arrive at/leave UGS ℓ with path information $\mathcal{P}$ and guarantee capture

however introducing variability in target speed would considerably increase the complexity in the resulting estimation problem. We have solved the related maximal delay problem for the scenario where the target speed always lies within known bounds (e.g., minimum and maximum speed limits on the road) (Krishnamoorthy et al. 2016a). Briefly, the UAV strategy in that setting is the following: when it waits at a node, it is optimal to assume that the target is going at minimum speed (so as to not miss the target) and when it moves, the UAV assumes that the target can (possibly) arrive at the next node at full speed—for details, see Krishnamoorthy et al. (2016a).

2.3 Evolution of information state

At decision epoch $t_k; k = 0, 1, \ldots$, let the UAV be at UGS location $\ell(p_k)$. The decision variable, $u_k \in \mathcal{U}$ indicates the node (on the time tree) that the UAV selects next. Even though the UAV and target motion evolve in continuous time, decisions are made (by the UAV only) at discrete time steps. The UAV makes a decision at time t_k after obtaining the measurement y_k. We stipulate that $y_k \in \{green, red\}$, with *red* indicating that the target visited UGS $\ell(p_k)$ at time $a(p_k)$ and *green* indicating the opposite. Suppose the target path information available to the UAV at time t_k is $\mathcal{P}_k$. Recall that the UAV starts at UGS 1 at time t_0. It follows that the initial path uncertainty information is given by:

$$\mathcal{P}_0 = \mathbf{P}((1, 0)) = \{1, \ldots, n\}. \tag{4}$$

The control action u_k is dependent on the current time, UAV position and most recent information: $u_k = \mathcal{F}(t_k, \ell(p_k), \mathcal{P}_k)$, where the mapping $\mathcal{F}$ is to be determined by an optimality principle—see (16) in the sequel. We shall refer to the tuple $s := (t, \ell(p), \mathcal{P})$ as the information state. The state evolves according to:

$$p_{k+1} = u_k,$$

$$t_{k+1} = \begin{cases} a(u_k) \text{ if } a(u_k) \geq t_k + d_p(\ell(p_k), \ell(u_k)), \\ t_k + d_p(\ell(p_k), \ell(u_k)), \text{ otherwise.} \end{cases} \tag{5}$$

$$\mathcal{P}_{k+1} = \begin{cases} \mathbf{P}(u_k), \ y_{k+1} = red, \\ \mathcal{P}_k \backslash \mathbf{P}(u_k), \ y_{k+1} = green. \end{cases} \tag{6}$$

Recall that $a(u_k)$ is the arrival time of the target at UGS location $\ell(u_k)$ and so, $a(u_k)$ is also the earliest possible time at which information is available at node u_k. So, if the UAV arrives at UGS $\ell(u_k)$ before that time, it shall wait until information becomes available at that UGS location. At decision

epoch t_{k+1}, the UAV receives the measurement y_{k+1} indicating whether or not the target has taken a path $\rho \in \mathbf{P}(u_k)$. Although $\mathcal{P}_{k+1}(u_k, y_{k+1})$ is an explicit function of its two arguments, we shall drop the dependance when the context is clear. The UAV never waits at (or revisits) an UGS location after receiving a "red" measurement from it—since the target never revisits any UGS location ("no cycles" assumption) and so the UAV gains no new information by doing so. To denote the time difference (which shall become critical later in the paper) between the decision epochs, we use:

$$\delta(s_k, u_k) = t_{k+1} - t_k = \max\{a(u_k) - t_k, d_p(\ell(p_k), \ell(u_k)\}. \tag{7}$$

Before we dive into the optimization problem, we define the regions of the state space that pertain to target capture and escape.

Capture condition: A sufficient condition for guaranteed capture is that the information state $s = (t, \ell(p), \mathcal{P})$ satisfy:

$$\mathcal{P} \subseteq \mathbf{P}(u), \text{ for some } u \in \mathcal{U} \text{ and}$$
$$t + d_p(\ell(p), \ell(u)) \leq a(u), \tag{8}$$

where $\mathbf{P}(u)$ is defined earlier—see (1). The above condition implies that the information available to the UAV is that all possible paths taken by the target must go through UGS $\ell(u)$ and furthermore, the UAV can travel to $\ell(u)$ before the target can get there, regardless of the path it chooses. We note that the above condition is also *necessary* for guaranteed capture in a single move by the UAV. Accordingly, let the capture set corresponding to state s be:

$$\mathcal{C}(s) = \{u \in \mathcal{U} : \mathcal{P} \subseteq \mathbf{P}(u), t + d_p(\ell(p), \ell(u)) \leq a(u)\}. \tag{9}$$

Escape condition: The current time $t > e(\rho)$ for some $\rho \in \mathcal{P}$, where $e(\rho)$ is the target arrival time at the exit node along path ρ. This is so because the current information available to the UAV indicates that the target has already exited the road network under a worst case scenario.

3 Optimization problem

Given the definitions earlier, we are now in a position to formally state the worst case minimum time capture problem. The UAV arrives at entry UGS 1 at time $t_0 > 0$ and is made aware of the target. One can easily visualize (from Fig. 1) that when t_0 is small, capture should be possible, given that the UAV has an advantage over the ground target—see (3). On the other hand, if t_0 is large, the target will likely escape, no matter what the UAV does.

Let the set of all information states be $\mathcal{S}$. We define a pursuit policy $\Pi : \mathcal{S} \to \mathcal{U}$ to be a mapping from the set of all information states to the set of nodes in the time tree G. The system state evolves over a finite time window $[0, T]$, where $T = \max_{\rho=1}^{n} e(\rho)$ is the maximum of all possible target exit times from the road network. This comes from the assumption that the target has unit speed and travels on a DAG with finite path lengths from the entry node to every exit node. The information state $s = (t, \ell(p), \mathcal{P})$, where $t \in [0, T]$, $p \in \mathcal{U}$ and $\mathcal{P}$ can be any non-empty subset of $\{1, \ldots, n\}$. Furthermore, t only takes discrete values from $[0, T]$ in increments of the UAV travel time (plus waiting time, if any). So, $\mathcal{S}$ is a countable finite set. For the target, the only action available is the path it chooses. Note that under our definition, a path from the entry UGS to an exit UGS fully specifies what turns the target must take at each intersection on the original road network (see Fig. 1a). For a given time tree G with $|\mathcal{U}|$ nodes, n possible target paths and initial delay t_0, we are interested in computing the pursuit policy, if it exists, that leads to capture in minimum time under worst-case target action. As such, we are only interested in computing the optimal control actions for the reachable portions of the state space starting from the initial condition, $s_0 = (t_0, 1, \{1, \ldots, n\})$.

3.1 Mathematical formulation

Let $\mathcal{M}(s_0)$ denote the worst-case minimum time to capture the target from the initial information state, $s_0 = (t_0, 1, \mathbf{P}(1, 0))$. To mathematically define $\mathcal{M}(s_0)$, we proceed in the following fashion. Let the target's chosen path be $\kappa \in \{1, \ldots, n\}$. Let $\pi = (u_0, \ldots, u_{\hat{m}})$ entail a finite sequence of non-repeating nodes that the UAV selects after leaving UGS 1. We only need to consider a finite sequence because of the finite terminal time assumption, i.e., the target if it is not captured, will leave the network in finite time. Inevitably, if the target is not captured, the escape condition will be met after a finite number of UAV moves. Furthermore, we only consider sequence of non-repeating nodes due to the DAG assumption i.e., the target does not go in cycles and hence, there is no value in the UAV revisiting a node. Since there are a maximum of $|\mathcal{U}| - 1$ distinct nodes left to select, we have the sequence length, $\hat{m} + 1 \leq |\mathcal{U}| - 1$.

The control action: $u_j(s_j)$, $j = 0, \ldots, \hat{m}$, is a function of the information state at decision epoch t_j. Furthermore, the observations: $y_j(\pi, \kappa)$, $j = 1, \ldots, \hat{m}$, made by the UAV are a function of both the UAV policy and target path. Since we are only interested in policies with a capture guarantee, it is necessary that capture occurs at UGS location $\ell(u_{\hat{m}})$ and time $a(u_{\hat{m}})$. This implies that the state at decision epoch $t_{\hat{m}}$ must satisfy the sufficient condition for guaranteed capture (8). In other words:

$$\mathcal{P}_{\hat{m}} \subseteq \mathbf{P}(u_{\hat{m}}) \text{ and}$$
$$t_{\hat{m}} + d_p(\ell(u_{\hat{m}-1}), \ell(u_{\hat{m}})) \leq a(u_{\hat{m}}). \tag{10}$$

The time to capture for the UAV policy and target path tuple (π, κ) is therefore given by:

$$J(\pi, \kappa) = a(u_{\hat{m}}) - t_0 = \sum_{i=0}^{\hat{m}} \delta(s_i, u_i)$$
$$= \delta(s_0, u_0) + \sum_{i=1}^{\hat{m}} \delta(s_i, u_i), \tag{11}$$

where the time difference between decision epochs $\delta(\cdot)$ is defined earlier (7). Let $\Pi(s_0)$ be the set of all sequences of non-repeating nodes of length $\hat{m}+1$ with a capture guarantee, starting from state s_0. So, we have the performance metric of interest given by:

$$\mathcal{M}(s_0) = \min_{\pi \in \Pi(s_0)} \max_{\kappa \in \mathcal{P}_0} J(\pi, \kappa). \tag{12}$$

From (11), we have:

$$\mathcal{M}(s_0) = \min_{\pi \in \Pi(s_0)} \max_{\kappa \in \mathcal{P}_0} \sum_{i=0}^{\hat{m}} \delta(s_i, u_i)$$
$$= \min_{u_0} \left\{ \delta(s_0, u_0) + \min_{u_1, u_2, \ldots} \max_{\kappa \in \mathcal{P}_0} \sum_{i=1}^{\hat{m}} \delta(s_i, u_i) \right\} \tag{13}$$
$$= \min_{u_0} \left\{ \delta(s_0, u_0) + \max_{y_1} \min_{\pi' \in \Pi(s_1)} \max_{\kappa' \in \mathcal{P}_1(u_0, y_1)} \sum_{i=1}^{\hat{m}} \delta(s_i, u_i) \right\} \tag{14}$$
$$\Rightarrow \mathcal{M}(s_0) = \min_{u_0 \in \mathcal{U}} \left\{ \delta(s_0, u_0) + \max_{y_1 \in Y(s_0, u_0)} \mathcal{M}(s_1) \right\}, \tag{15}$$

where $Y(s_0, u_0) \subseteq \{red, green\}$ is the set of possible observations at UGS $\ell(u_0)$. In (13), we replace the minimization over the policy π with incremental optimization via selection of the optimal control variables: $u_i; i = 0, \ldots, \hat{m}$. Also, we can pull out $\delta(s_0, u_0)$ from the inner maximization because, by definition (7), it only depends on the choice of control variable: u_0. For the target, choosing an optimal path κ is equivalent to sequentially maximizing $J(.,.)$ over the set of likely observations seen at UGS locations $\ell(u_i); i = 0, \ldots, \hat{m}$. Furthermore, the first observation, y_1 is not dependent on future UAV actions, u_1, u_2, etc. Hence the maximization with respect to y_1 can be pulled out of the inner minimization over future control actions in (14). In (15), the path information, $\mathcal{P}_1(u_0, y_1)$ is updated according to (6). By definition, $\mathcal{M}(s_1)$ is the worst case minimum time to capture from the information state, s_1. Hence, we have a recursive equation (15) for computing the performance metric of interest. We acknowledge that the recursion (15) bears similarity to the forward dynamic programming recursion for sequential decision making under uncertainty derived in LaValle (2006)—specifically, see min–max recursion (10.39) in chapter 10 of the book. Note that it is entirely possible that the set $\Pi(s_0) = \emptyset$. In other words, capture is not possible from the initial state, s_0. We shall address this case in the next section, where we enumerate how the recursion will be solved.

3.2 Min–max optimization

The minimum time to capture from state s_k satisfies the recursion:

$$\mathcal{M}(s_k) = \min_{u_k \in \mathcal{U}} \left\{ \delta(s_k, u_k) + \max_{y_{k+1} \in Y(s_k, u_k)} \mathcal{M}(s_{k+1}) \right\}. \tag{16}$$

In (16), the state at the next decision epoch s_{k+1} is updated according to (5) and (6). We shall refer to (16) simply as the Recursive Equation (RE). Note that by definition, the recursion and the associated tree search is combinatorial. It is important to note that RE is not a standard dynamic programming equation. Since we do not know, a priori, how many steps it takes for the recursion to terminate, one must do a forward search until one of two boundary conditions corresponding to either capture or escape is encountered. We reiterate that although the action space is finite (set of all possible nodes in the time tree), high branching factor in the tree may lead to poor scaling. Indeed, the problem under consideration has been shown to be NP-hard (for details on complexity, see Sundaram et al. 2017). We will show shortly why it is sufficient to consider no more than n moves for the UAV. A naive exhaustive search would entail searching over all possible pursuer actions (non-repeating sequences of length n starting at node $(1, 0)$) and all possible target paths and so the search space is of size:

$$\prod_{k=1}^{n} (|\mathcal{U}| - k) \times n. \tag{17}$$

Clearly, any reduction in the search space would be useful. Towards this end, we provide a series of Lemmas that bring out key properties of the optimal control policy, eventually leading to a reduced computational burden.

Lemma 1 *If $\mathcal{C}(s)$ is non-empty, the minimum time to capture from state s is given by:*

$$\mathcal{M}(s) = \min_{u \in \mathcal{C}(s)} a(u) - t \tag{18}$$

Let u^* be the minimizing control action in (18). The UAV proceeds to $\ell(u^*)$ and waits until the target gets there.

Lemma 2 *If the current time $t > e(\rho)$ for some $\rho \in \mathcal{P}$, then the target would escape (in the worst-case) and so,*

$$\mathcal{M}(s) = \infty, \ \text{if } t > e(\rho) \text{ for some } k \in \mathcal{P}. \tag{19}$$

Lemma 3 *The minimizing control, u, to RE (16) is such that: $red \in Y(s_k, u)$.*

Proof Suppose $red \notin Y(s_k, u)$, i.e., the only possible observation at $\ell(u)$ is a green UGS. This implies that $\mathcal{P}_k \cap \mathbf{P}(u) = \emptyset$. In other words, there is no information to be gained and hence, no reduction in uncertainty possible by visiting $\ell(u)$. Rather, the UAV will incur an additional travel cost of $d_p(\ell(p_k), \ell(u))$. From the triangle inequality (2), it follows that the UAV is better off skipping $\ell(u)$. □

In light of the triangle inequality, we have shown that Lemmas 1 and 3 exclude some control actions as being sub-optimal. The UAV will select u only if one of two things happen. Either capture occurs at $\ell(u)$ or the path uncertainty is reduced at u. We are now in a position to formally define the optimality preserving restriction to the action space. Let the restricted action space:

$$\mathcal{B}(\mathcal{P}) = \{u \in \mathcal{U} : \mathcal{P} \cap \mathbf{P}(u) \neq \emptyset\}. \tag{20}$$

Given Lemmas 1–3, without any loss in optimality, we can rewrite RE as follows:

$$\mathcal{M}(s_k) = \begin{cases} \infty, \ \text{if } t > e(\rho) \text{ for some } \rho \in \mathcal{P}_k, \\ \min_{u \in \mathcal{C}(s)} \left[a(u) - t_k\right], \ \mathcal{C}(s) \neq \emptyset, \\ \min_{u \in \mathcal{B}(\mathcal{P}_k)} \left[\delta(s_k, u) + \right. \\ \left. \max_{y \in Y(s_k,u)} \mathcal{M}(s_{k+1})\right], \ \mathcal{C}(s) = \emptyset. \end{cases} \tag{21}$$

The revised RE (21) deals with three cases:

1) imminent escape: capture is not possible.
2) imminent capture: capture occurs at the (earliest possible) UGS corresponding to a node in the capture set.
3) path uncertainty reduction: the time to capture is determined by a min–max recursion.

The escape and capture *boundary conditions* for which the solution is immediate and no further recursion is required are given by 1) and 2). For case 3), the recursion relies on solutions to the sub-problems corresponding to all $u \in \mathcal{B}(\mathcal{P}_k)$ and $y \in Y(s_k, u)$. Note however that the size of the original problem is a function of the cardinality of $\mathcal{P}_k$. So, the *size* of the sub-problems to be solved for each choice of (u, y) is reduced since the uncertainty in the path is reduced for this case. The solution to the min–max recursion yields the optimal sequence of pursuer actions and corresponding worst case target path. Moreover, we also obtain a pursuit policy, i.e., the optimal action for all the reachable information states from the initial state, s_0. This is so because, the max operation requires the computation of cost to go for both observations (red and $green$) at each decision epoch and hence, we end up computing the best (UAV) response for both. This process continues all the way down to the leaf nodes in the time tree.

Note that the optimal pursuer action is constrained to enforce a reduction in target path uncertainty at every move. Indeed the cardinality of the path uncertainty set will reduce, at least by 1, for every move made by the UAV. As a result, the recursion will terminate in no more than n steps/moves! Recall that n is the number of paths that the target can take. In the worst case, the RE would entail exploring all possible pursuer action sequences and target paths—see (17) for size of the worst-case (exhaustive) search space.

3.3 Tightening escape boundary condition

One can achieve additional savings if the search paths that lead to escape can be identified sooner. Indeed, let $\mathcal{D}(\ell|\mathcal{P})$ be the latest time the UAV can arrive at/leave UGS ℓ and guarantee capture, armed with the path information $\mathcal{P}$. In prior work (Krishnamoorthy et al. 2016b), the authors have established a backward dynamic programming approach to compute $\mathcal{D}(\ell|\mathcal{P})$ under an identical problem setup as considered in this paper. For the maximal delay setup, the optimization entails a $max\ min$ recursion similar in flavor to the RE (16) but with one crucial difference. The optimal recursion for a given information set depends only on solutions to information sets of lower cardinality. Hence, the maximal delays at each node for different path information sets can be computed in the order of increasing cardinality of the sets, for details see Krishnamoorthy et al. (2016b). Indeed, for a given instance of the problem, $\mathcal{D}(\ell|\mathcal{P})$ can be computed $\forall \ell = 1, \ldots, m$ and $\forall \mathcal{P}$. Note that, in general, the path uncertainty set can be any proper subset of $\{1, \ldots, n\}$. If the current state satisfies: $t_k > \mathcal{D}(\ell(p_k)|\mathcal{P}_k)$, then the target would escape (in the worst-case) since the current time has exceeded the latest arrival time at $\ell(p_k)$ with a capture guarantee. And so, one can tighten the *boundary condition* for escape as follows:

$$\mathcal{M}(s_k) = \infty, \ \text{if } t_k > \mathcal{D}(\ell(p_k)|\mathcal{P}_k). \tag{22}$$

The above modification is only helpful if the additional computation time involved in pre-computing $\mathcal{D}(\ell|\mathcal{P}), \forall \ell, \mathcal{P}$ is small. The algorithm provided in Krishnamoorthy et al. (2016b) for computing the required maximal delays has complexity $\mathcal{O}(2^n m \log m)$. Note that the values can be pre-computed once and stored in memory. They can be used as and when required during implementation of the revised recursion RE (21). We shall show later using a real world example that the benefits of pre-computing the maximal delays can be significant.

3.4 Branch and bound search

Suppose we are in the midst of computing $\mathcal{M}(s_0)$ using the revised RE (21) and $\mathcal{C}(s_0) = \emptyset$ (else, the search is trivial and terminates in one step). Further suppose we have already computed a feasible solution. Indeed, let the current best solution, i.e., a pursuit policy $\pi = (u_0, \ldots)$ with the lowest cost (so far) be given by $\mathbf{C}_\pi < \infty$. Let $\bar{\Pi}$ denote all possible pursuit policies such that the initial control action, $\bar{u} \neq u_0$. For this choice of control action $\bar{u}$ and any observation y, we have the associated cost given by:

$$\mathcal{M}_{\bar{u},y}(s_0) = \delta(s_0, \bar{u}) + \mathcal{M}(s_1), \tag{23}$$

where $s_1 = (t_1, \ell(\bar{u}), \mathcal{P}_1(\bar{u}, y))$ is updated according to (5) and (6). Suppose we can quickly compute a lower bound, $\underline{M}_y(\bar{\Pi})$ to $\mathcal{M}(s_1)$.

$$\begin{aligned} &\text{If } \delta(s_0, \bar{u}) + \underline{M}_y(\bar{\Pi}) \geq \mathbf{C}_\pi, \ \forall y \in \{red, green\}, \\ &\qquad \Rightarrow \mathcal{M}_{\bar{u},y}(s_0) \geq \mathbf{C}_\pi, \end{aligned} \tag{24}$$

and we infer that no policy $\bar{\pi} \in \bar{\Pi}$ can have a cost lower than $\mathbf{C}_\pi$. Hence, we can discard the entire set of policies $\bar{\Pi}$ from further consideration. This could result in potentially huge computational savings by eliminating the need to compute the recursions (21) for all possible $\bar{u}$, y. A prerequisite for the savings is that we are able to compute tight lower bounds quickly. We provide a recipe below for computing lower bounds for the two possible observations, $y \in \{red, green\}$. In either case, the target can be captured at any node $w \in \mathcal{C}(s_1)$, in the capture set for s_1. So, we have the lower bound:

$$\underline{M}_y(\bar{\Pi}) = \max_{w \in \mathcal{C}(s_1)} a(w) - t_1.$$

Note that if no such w exists for either y, we can always use the trivial lower bound: $\underline{M}_y(\bar{\Pi}) = 0$. In other words, we use the capture time that would be achieved if the target was captured at $\bar{u}$ itself. Upon computing the lower bound, we proceed with the recursion to compute $\mathcal{M}(s_1)$ only if $\delta(s_0, \bar{u}) + \underline{M}_y(\bar{\Pi}) < \mathcal{C}_\pi$. It is important to reiterate that neither of the two pruning methods detailed above result in loss of optimality.

4 Example problem

In this section, we shall demonstrate the solution method with numerical examples. First, we consider the simple illustrative example considered earlier from Fig. 1a. Then, we consider a more elaborate real world example which clearly demonstrates the savings in computational burden yielded by the branch and bound technique and pre-computation of the maximal delays at each node.

4.1 Numerical example

We shall use the revised RE (21) to compute the optimal pursuit policy and worst case target action for the example problem illustrated in Fig. 1a. We assume that the UAV travels between any two UGSs at the constant ground speed, $V = 1.62$. For this example, we use the revised RE (21) as is, since the problem size is small, i.e., $|\mathcal{U}| = 8$ and $n = 4$ and so, an exhaustive search would entail checking 3360 different combinations of UAV action sequences and target paths. We shall illustrate the benefits of the pruning methods in the next section, where we deal with a real world example. For an initial delay, $t_0 = 4$ at UGS 1, the optimal control policy and cost to go are shown in Fig. 2. In the main plot (shown as a decision graph), we indicate the UGS location followed by the time to capture from that location indicated by $\mathcal{M}$. At capture locations, we also indicate the waiting time, if any, by $\mathcal{W}$. So, the worst case minimum time to capture from UGS 1, $\mathcal{M}(t_0 = 4) \approx 13.537$. Also shown is the pursuit policy for all reachable information states. For instance, the optimal UAV action from UGS 1 is to go to UGS 2. If UGS 2 is red (shown in bold red), the UAV goes to UGS 7 and waits for $\mathcal{W} \approx 2.205$ time units. If UGS 2 is green instead (shown in dotted green), the UAV goes to UGS 5 and so on, until capture occurs in the worst-case at UGS 7. Also, in the inset of Fig. 2, we show the worst case target path, $1 \to 3 \to 4 \to 7$ (as dashed orange arrows) and corresponding UGS visit sequence for the UAV (via blue arrows). If we reduce the initial delay to $t_0 = 2$, the minimum time to capture from UGS 1, $\mathcal{M}(t_0 = 2) \approx 12.657$. From Fig. 3, we see that the target is captured, in the worst-case, again at UGS 7. However, the worst-case target path for $t_0 = 2$ is $1 \to 2 \to 7$ (see inset of Fig. 3). Since this

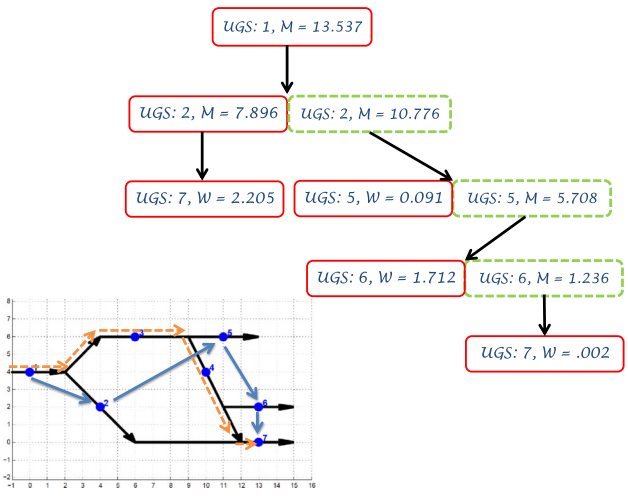


Fig. 2 Decision tree and time to capture for $t_0 = 4$

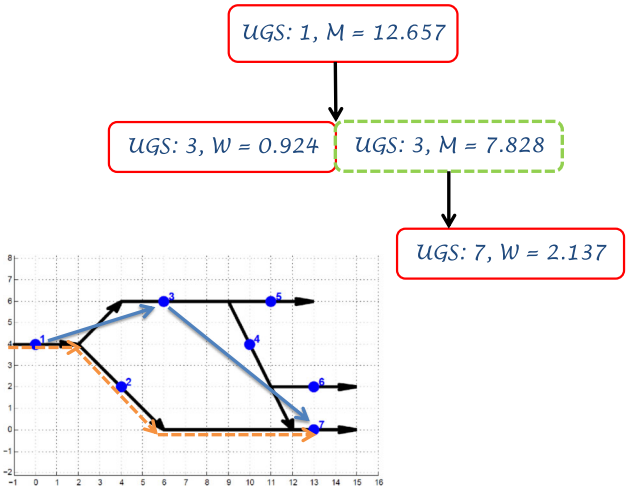


Fig. 3 Decision tree and time to capture for $t_0 = 2$

path is shorter than the earlier one for $t_0 = 4$, the target arrives earlier at UGS 7. The capture time is given by the difference between the target arrival time at the capture location and the initial delay. For this example, it turns out that the time to capture for $t_0 = 2$ is shorter than $t_0 = 4$ case although this is not necessarily true in general. In other words, the capture time is not necessarily monotonic increasing with the initial delay as shown by a counterexample: consider a simple graph containing exactly one path connecting the entry UGS to an exit UGS. For this scenario, the capture time will decrease as the initial delay increases (up to a point) simply because the UAV has to spend a shorter time at the exit UGS waiting for the target to show up.

4.2 Real world example

We tested the proposed algorithm on a real road network testbed located in Camp Atterbury, IN, USA shown in Fig. 4. The road network spread over a 3 km × 2 km rectangular area constitutes 12 UGSs (shown in red) distributed along roads marked in yellow shown in a Google earth satellite image. The target is a ground vehicle traveling along the roads at a constant speed of 20 miles per hour (mph) and the pursuer is a fixed wing UAV traveling at a fixed ground speed of 40 mph. The rules of the road i.e., allowed turns at each intersection, have been chosen such that there are no cycles. As before, the UAV can travel between any two UGS locations. There are 15 different paths that the target can travel along; starting from the entry UGS 5 and ending in one of the exit UGSs: 16, 22 and 23.

We assume that the target enters the road network via UGS 5 at time 0. So, the time tree showing UGSs along different paths and the corresponding time of arrival (in minutes) is shown in Fig. 5. For this real world example, we have $n = 15$

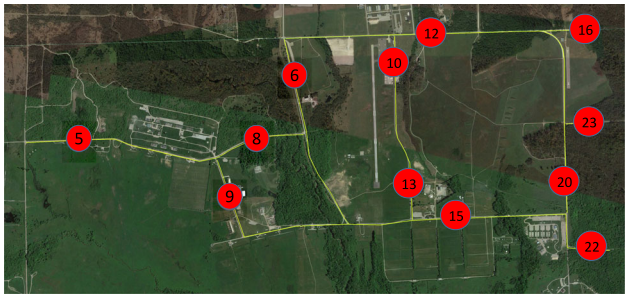


Fig. 4 Road network with UGSs in Camp Atterbury, IN, USA

and $|\mathcal{U}| = 37$ making it an unmanageable search space of $\mathcal{O}(10^{23})$ states.

We compute the minimum time capture policy using the revised RE (21) for different initial delays seen at the entry UGS 5. We also implement the maximal delay and branch and bound pruning methods detailed in Sects. 3.3 and 3.4. To implement the former, we first compute the maximal delay with a capture guarantee at UGS 5. For this network, we compute this to be $\mathbf{d} = \mathcal{D}(5|\{1, \ldots, 15\}) \approx 5.64$ min. Note that as a by-product, we also get the maximal delays for all possible information states at every UGS location—for details, see Krishnamoorthy et al. (2016b). Notably, the computation time required for this pre-processing step was less than a second.

In Fig. 6, we illustrate the benefits of the pruning methods in reducing the computational burden for all initial delays $t_0 \in \{1, 2, 3, 4, 5, \mathbf{d}\}$. In this plot, RE, RE (B and B) and RE (MD) stand for the revised RE method (21), revised RE plus the branch and bound technique and revised RE with the pre-computed maximal delay data (22) respectively. Finally, RE (BBMD) includes both pruning methods combined. For a given t_0, the time to capture for all four methods are identical and equal to the minimum time to capture, $\mathcal{M}(t_0)$. In other words, there is no loss in optimality in incorporating the pruning methods. For the revised RE, the computation time initially increases with t_0 since a bigger portion of the time tree in Fig. 5 is considered in the search. However, as we get closer to the maximal delay, a number of search paths lead to infeasibility (escape condition being met). As mentioned earlier, we see two orders of magnitude reduction in computation time across the board for different initial delays.

To understand the optimal pursuit policy, we focus on initial delay, $t_0 = 2$ min at UGS 5. For this case, the optimal control policy and cost to go for all reachable information states from s_0 are shown in Fig. 7. The worst-case minimum time to capture from UGS 5, $\mathcal{M}(t_0 = 2) \approx 7.593$ min. For this case, the worst-case target path is given by $5 \rightarrow 8 \rightarrow 13 \rightarrow 10$. In the inset of Fig. 7, we show the corresponding sequence of optimal UAV actions (blue arrows). Accordingly, the UAV sees the following observa-

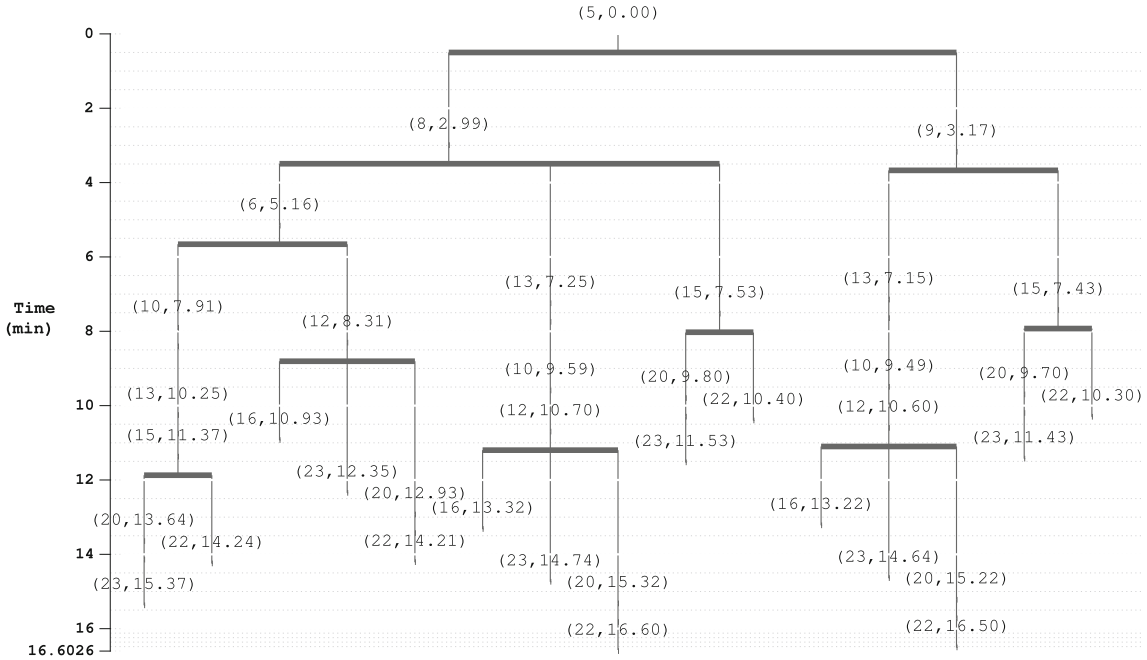


Fig. 5 Time tree showing nodes (UGS location and target time of arrival)

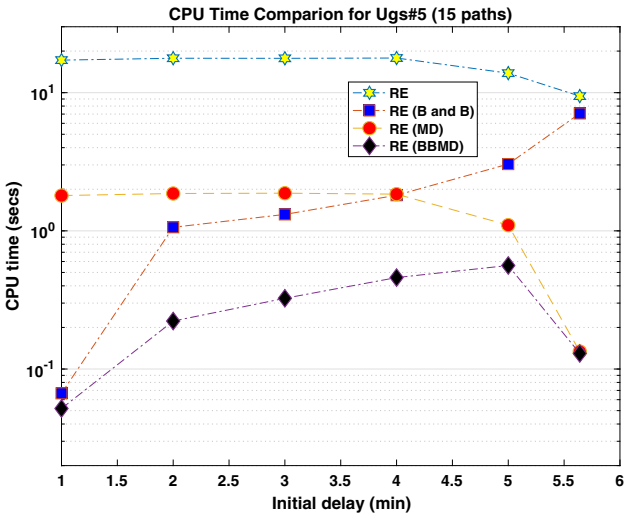


Fig. 6 CPU time comparison for 15 possible target paths

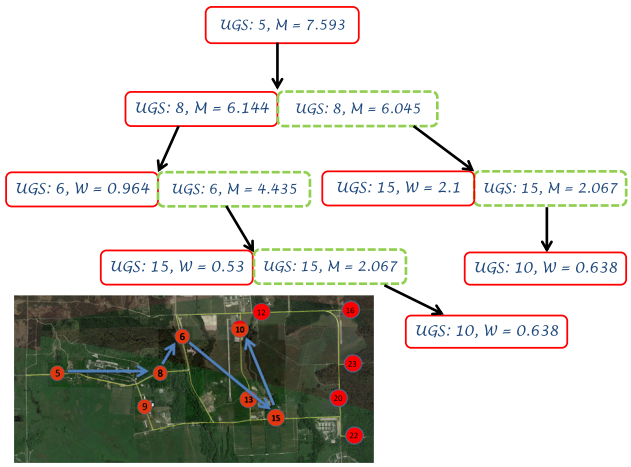


Fig. 7 Decision tree and time to capture for $t_0 = 2$ min

tions: 5 (red), 8 (red), 6 (green), 15 (green) and finally 10 (red) indicating capture. Along the way, the UAV infers that the target turned south after leaving UGS 8 and turned north at the intersection leading to UGSs 13 and 15.

We observed similar savings in computational burden for different configurations of the real world road network (e.g., by changing the traffic rules as to what turns are allowed and thereby changing the number of allowed paths n and/or degree of the time tree). Figure 8 shows the CPU time comparison, as before, for a larger problem with $n = 25$ possible target paths. In this case, we clearly see the benefits of the pruning methods which bring the otherwise exorbitant computation time into the realm of real time implementation. All computation times reported in this work correspond to Matlab code run times on a MacBook Pro with 3.1 GHz Intel Core i7 processor and 16 GB memory.

5 Conclusions

The optimal control of a UAV tasked with intercepting a ground target on a road network instrumented with UGSs is considered. The UAV is interrogating the UGSs, some of which were triggered by the moving target, and as such has

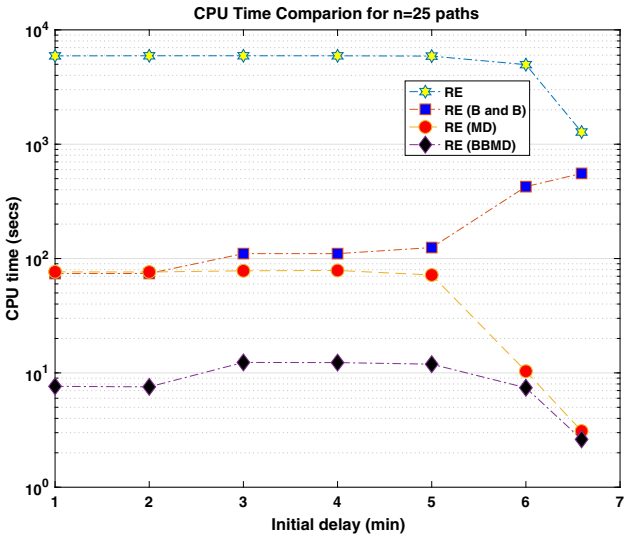


Fig. 8 CPU time comparison for 25 possible target paths

access only to partial observations of the system state. The resulting deterministic pursuit-evasion game on a finite time tree where the target chooses a path and the UAV has partial information, is solved using a forward search min–max recursion. To reduce the computational burden, we pre-compute the maximal arrival times for the UAV with a capture guarantee for all UGSs in the network and thereby tighten the terminal condition for intruder escape. We also employ a branch and bound technique that demonstrates significant savings in computational time when applied to a real world problem instance. Future work shall focus on relaxing the error free observation model. We are aware of the significant obstacles one would encounter in solving the relaxed problem. It is quite likely that one has to resort to approximate solutions with probabilistic bounds on time to capture. One other promising research direction would be to enable the ground target to take evasive actions based on awareness about the UAV and UGS locations.

References

Başar, T. (2001). *Control theory: Twenty-five seminal papers (1932–1981), chap. dual control theory* (pp. 181–196). New York: Wiley-IEEE Press.

Borie, R., Tovey, C., & Koenig, S. (2011). Algorithms and complexity results for graph-based pursuit evasion. *Autonomous Robots*, *31*(4), 317. https://doi.org/10.1007/s10514-011-9255-y.

Carlone, L., Axelrod, A., Karaman, S., & Chowdhary, G. (2018). *Handbook of dynamic data driven applications systems, chap. aided optimal search: Data-driven target pursuit from on-demand delayed binary observations* (pp. 295–335). Cham: Springer.

Chen, H., Krishnamoorthy, K., Zhang, W., & Casbeer, D. (2017). Intruder isolation on a general road network under partial information. *IEEE Control Systems Technology*, *25*(1), 222–234. https://doi.org/10.1109/TCST.2016.2550423.

Chung, T. H., Hollinger, G. A., & Isler, V. (2011). Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, *31*(4), 299–316.

Clarke, N. E. (2009). A witness version of the cops and robber game. *Discrete Mathematics*, *309*(10), 3292–3298.

Clarke, N. E., Cos, D., Duffy, C., Dyer, D., Fitzpatrick, S., & Messinger, M. E. (2017). *Limited visibility cops and robbers*. arXiv:1708.07179

Demirbas, M., Arora, A., & Gouda, M. (2003). *Self-stabilizing systems, lecture notes in computer science, vol. 2704, chap. A pursuer-evader game for sensor networks* (pp. 1–16). Berlin: Springer.

Dereniowski, D., Dyer, D., & Tifenbach, R. (2015). Zero-visibility cops and robber and the pathwidth of a graph. *Journal of Combinatorial Optimization*, *29*(3), 541–564.

Dumitrescu, A., Kok, H., Suzuki, I., & Zylinski, P. (2010). Vision-based pursuit-evasion in a grid. *SIAM Journal of Discrete Mathematics*, *24*(3), 1177–1204.

Dzyubenko, G. T., & Pshenichnyi, B. N. (1972). Discrete differential games with information lag. *Cybernetics and Systems Analysis*, *8*(6), 947–952.

Fomin, F. V., & Thilikos, D. M. (2008). An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, *399*(3), 236–245.

Isler, V., & Karnad, N. (2008). The role of information in cop-robber game. *Theoretical Computer Science*, *399*(3), 179–190.

Kehagius, A., Mitsche, D., & Pralat, P. (2014). *The role of visibility in pursuit/evasion games*. arXiv:1402.6136

Krishnamoorthy, K., Casbeer, D., & Pachter, M. (2016a). Pursuit of a moving target with bounded speed on a directed acyclic graph under partial information. *IMA Journal of Mathematical Control and Information*, *32*, 1–16. https://doi.org/10.1093/imamci/dnw061.

Krishnamoorthy, K., Casbeer, D., & Pachter, M. (2016b). Pursuit of a moving target with known constant speed on a directed acyclic graph under partial information. *SIAM Journal of Control and Optimization*, *54*(5), 2259–2273. https://doi.org/10.1137/140994216.

Krishnamoorthy, K., Casbeer, D. W., & Pachter, M. (2015). Minimum time UAV pursuit of a moving ground target using partial information. In *International conference on unmanned aircraft systems (ICUAS)* (pp. 204–208). Denver, CO.

Krishnamoorthy, K., Darbha, S., Khargonekar, P., Casbeer, D. W., Chandler, P., & Pachter, M. (2013a). Optimal minimax pursuit evasion on a Manhattan grid. In *American control conference* (pp. 3427–3434). Wasington, DC.

Krishnamoorthy, K., Darbha, S., Khargonekar, P., Chandler, P., & Pachter, M. (2013b). Optimal cooperative pursuit on a Manhattan grid. In *AIAA guidance, navigation and control conference, AIAA 2013-4633*. Boston, MA

LaValle, S. M. (2006). *Planning algorithms, chap. sequential decision theory*. Cambridge: Cambridge University Press. http://planning.cs.uiuc.edu/ch10.pdf.

Oh, Y., Choi, S., & Oh, S. (2018). Chance-constrained target tracking using sensors with bounded fan-shaped sensing regions. *Autonomous Robots*, *42*(2), 307–327. https://doi.org/10.1007/s10514-017-9656-7.

Parsons, T. D. (1978). *Pursuit-evasion in a graph, lecture notes in mathematics* (vol. 642, pp. 426–441). Berlin: Springer.

Rasmussen, S., Krishnamoorthy, K., & Kingston, D. (2016). Field experiment of a fully autonomous multiple UAV/UGS intruder detection and monitoring system. In *International conference on unmanned aircraft systems* (pp. 1293–1302). Arlington, VA. https://doi.org/10.1109/ICUAS.2016.7502563.

Robin, C., & Lacroix, S. (2016). Multi-robot target detection and tracking: Taxonomy and survey. *Autonomous Robots*, *40*(4), 729–760. https://doi.org/10.1007/s10514-015-9491-7.

Sinopoli, B., Sharp, C., Schenato, L., Schaffert, S., & Sastry, S. (2003). Distributed control applications within sensor networks. *Proceedings of the IEEE*, *91*(8), 1235–1246.

Sugihara, K., & Suzuki, I. (1989). Optimal algorithms for a pursuit-evasion problem in grids. *SIAM Journal of Discrete Mathematics*, *2*(1), 126–143.

Sundaram, S., Krishnamoorthy, K., & Casbeer, D. (2017). Pursuit on a graph under partial information from sensors. In *American control conference* (pp. 4279–4284). Seattle, WA.

Svec, P., & Gupta, S. K. (2012). Automated synthesis of action selection policies for unmanned vehicles operating in adverse environments. *Autonomous Robots*, *32*(2), 149–164. https://doi.org/10.1007/s10514-011-9268-6.

Vieira, M. A. M., Govindan, R., & Sukhatme, G. S. (2009). Scalable and practical pursuit-evasion with networked robots. *Intelligent Service Robotics*, *2*(4), 247–263.

Dr. Krishna Kalyanam received the B. Tech. degree in Mechanical engineering from the Indian Institute of Technology, Madras in 2000, and the M.S. and Ph.D. degrees in Mechanical engineering from the University of California at Los Angeles, in 2003 and 2005, respectively. He is a Sr. Scientist with the System Sciences Lab at the Palo Alto Research Center (PARC). Prior to that, he was a research scientist and in-house contractor at the Control Science Center of Excellence, Air Force Research Laboratory. His research interests include autonomous systems, optimization and sequential decision making.

Dr. David Casbeer is the Team Lead for the UAV Cooperative and Intelligent Control Team in the Control Science Center of Excellence, Aerospace Systems Directorate, Air Force Research Laboratory. In this capacity he leads a team of researchers investigating the cooperative control of autonomous UAVs with a particular emphasis on high-level decision making and planning under uncertainty. Dr. Casbeer received BS and Ph.D. degrees from Brigham Young University in 2003 and 2009. His current work involves GPS-denied cooperative navigation, UAV self-protection, and ground intruder tracking by UAV/UGS. He currently serves as the Chair for the AIAA Intelligent Systems Technical Committee and is a Senior Editor for the Journal of Intelligent and Robotic Systems.

Dr. Meir Pachter is a Professor of Electrical Engineering at the Air Force Institute of Technology, Wright-Patterson AFB. Dr. Pachter received the BS and MS degrees in Aerospace Engineering in 1967 and 1969 respectively, and the Ph.D. degree in Applied Mathematics in 1975, all from the Israel Institute of Technology. His current areas of interest include military operations optimization, dynamic games, cooperative control, estimation and optimization, statistical signal processing, INS and GPS navigation.