Multiple Pursuers Under Partial Information from Sensors
Blake Wilson, Amritha Prasad, Cheng-Kok Koh, and Shreyas Sundaram

Abstract—We consider a pursuit evasion scenario on a road network represented by a directed acyclic graph where a set of pursuers seek to capture an evader before it reaches any of the terminal nodes. Each node of the graph represents a ground sensor which is tripped if the evader passes through it. Pursuers do not share information and each must travel to a ground sensor to acquire its information. We show that it is NP-Hard to determine the maximum delay for multiple pursuers to guarantee capture of the evader. Therefore, we construct a class of pursuit policies using a new vertex separator called ‘Sweep-Covers’ to reduce the computational complexity. Finally, we study the effect of increasing the number of ‘Sweep-Covers’ to reduce the computational complexity. We show that it is NP-Hard to determine the maximum delay for sending partial information from sensors. A previous study on this game by Krishnamoorthy et al. [7] gave an exponential time algorithm for finding the optimal policy for a single pursuer to capture an evader. Previous work by Sundaram et al. [8] restricted pursuer paths to a node-sweeping policy for an approximate solution. We will consider a similar solution that expands the node-sweeping policy to a group-sweeping policy that assigns induced sub-graphs of the road network to pursuers to sweep as a group. When a group of pursuers arrive at a node they can choose to diverge from one another to sweep a different sub-graph of the road network.

I. INTRODUCTION

Beginning with the original, graph constrained formulation of Pursuit-Evasion by Parsons [1], the game of capturing evaders in road networks has undergone many variants and developed a rich history. Some variations of the game impose road network constraints, e.g. Manhattan Grids [2], while others impose information constraints to the pursuers in the form of alarms [3], witness systems [4], proximity detection [5][6], and partial information from sensors [7][8]. Unlike other Pursuit-Evasion games where the pursuers can see the evader [9] [10][11][12][13] we consider a game where the pursuers are constrained to checking sensors within a road network for the evaders path [2][14], i.e., under partial information from sensors. Specifically, this paper will look at calculating the maximum delay for sending in a set of pursuers such that the pursuers are guaranteed to capture the evader while under partial information from sensors. A previous study on this game by Krishnamoorthy et al. [7] gave an exponential time algorithm for finding the optimal policy for a single pursuer to capture an evader. Previous work by Sundaram et al. [8] restricted pursuer paths to a node-sweeping policy for an approximate solution. We will consider a similar solution that expands the node-sweeping policy to a group-sweeping policy that assigns induced sub-graphs of the road network to pursuers to sweep as a group. When a group of pursuers arrive at a node they can choose to diverge from one another to sweep a different sub-graph of the road network.

A. Pursuit-Evasion Problem

Consider a road network defined by a directed acyclic graph (DAG), \( R = \{V_R, E_R\} \). This road network contains three types of nodes, a single source node \( s \in V_R \), a set of goal nodes denoted by \( G \subset V_R \), and a set of core nodes which compose the rest of the nodes, denoted by \( C \subset V_R \). Without loss of generality, we assume \( s \) has no incoming edges, and each goal node has no outgoing edges. A set of unmanned aerial vehicles (pursuers), denoted \( P \), and a ground vehicle (evader) enter the road network through \( s \). The evader enters at the time \( t = 0 \) whereas the pursuers enter at a time \( t = D \) for some \( D \geq 0 \). This entrance time for the pursuers will be known as the pursuer delay.

The evader can only travel along edges in the road network and the time taken by the evader to travel from \( v_i \) to \( v_j \) in \( R \) is denoted \( t_e(v_i, v_j) \). However, any pursuer is able to travel between any two nodes regardless of whether there is an edge between them. The time taken for any pursuer to travel from \( v_i \) to \( v_j \) in \( R \) is denoted \( t_p(v_i, v_j) \). All pursuers have the same speed advantage over the evader, \( \forall (v_i, v_j) \in E_R \) : \( t_p(v_i, v_j) \leq t_e(v_i, v_j) \). These pursuer travel times are symmetric, non-negative, and satisfy the triangle inequality i.e. \( \forall (v_i, v_j, v_k) \in R \) : \( t_p(v_i, v_j) \leq t_p(v_i, v_k) + t_p(v_k, v_j) \).

Each node within the graph, \( v_i \in V_R \), has an Unattended Ground Sensor (UGS) at the same location. Each UGS has two states: a green state, meaning that the evader has not passed through the node, and a red state, meaning the evader has passed through the node. The red state will also have a time stamp of when the evader passed through the node. The objective of the evader is to reach any of the goal nodes in \( G \) without being captured by any of the pursuers. The objective of the
pursuers is to capture (have at least one of the pursuers on the same node at the same time as the evader) before or exactly at the same time as the evader reaches a goal node. The only information available to any pursuer is the graph and the information obtained from the ground sensors. In this paper, we assume the pursuers do not share sensor information with other pursuers. A pursuer must reach a node, \( v_i \in V_R \), to obtain the state of the UGS. If it is red, it obtains the time stamp.

For each pursuer \( p \in P \), let \( H_p(t) \) be the history of the nodes visited by pursuer \( p \) up to time \( t \), along with the information received from the UGSs at each node in \( H_p(t) \). Note that \( \forall p \in P, H_p(t) = \emptyset \) for \( t < D \) and \( H_p(D) = s \). We will consider a policy for a pursuer \( p \in P \) to be a mapping \( \mu_p \) from the current location of the pursuer and its history \( H_p(t) \) to the next node the pursuer should visit. Note that we allow different \( \mu_p \) for different pursuers.

**Definition 1.** An instance of the Pursuit Evasion problem is given by a Directed Acyclic Graph (DAG), \( \mathcal{R} = \{V_R, E_R\} \), with a single source node \( s \), a non-negative evader travel time \( t_{v_i}(v_j) \) for each edge \( (v_i, v_j) \in E_R \), and a non-negative pursuer travel time \( t_p(v_i, v_j) \) for all distinct pairs of vertices \( v_i, v_j \in V_R \). The pursuer travel times are symmetric, satisfy the triangle inequality, and \( \forall (v_i, v_j) \in E_R : t_p(v_i, v_j) \leq t_e(v_i, v_j) \). Finally, the instance also consists of a positive integer \( k \) denoting the number of pursuers.

**Problem 1.** Maximum Pursuer Delay Problem (MPDP). Given an instance of the Pursuit Evasion problem, find the largest time \( D \) at which the pursuers can enter the graph so that there is a policy that guarantees capture of the evader.

**Problem 2.** Capture Feasibility Problem (CFP). Given an instance of the Pursuit Evasion problem, is there some time \( D > 0 \) at which the pursuers can enter the graph and still be guaranteed to capture the evader?

It is important to note that a solution to MPDP gives a solution to CFP. It was shown in [8] that MPDP for a single pursuer is NP-Hard on a class of trees known as “Spider Networks”. In addition, it was also shown that there does not exist an approximate solution to MPDP within a constant factor of the optimal solution. Using this result, it is simple to show that for any given number of pursuers \( k \), that MPDP and CFP are NP-Hard and they are inapproximable. Construct a tree such that \( k \) spider networks are joined together by a common source node. Then, increase the pursuer travel time between these spider networks so that it is impossible for a pursuer to be able to travel to 2 different spider networks before the evader has escaped the road network. Then, the only feasible policy is to assign each pursuer a spider network and solve MPDP on its respective spider network. This reduces to solving MPDP for a single pursuer on a spider network which is NP-Hard and inapproximable.

**II. Computing Maximum Pursuer Delay for a Class of Group-Sweeping Policies**

Due to MPDP being NP-Hard and inapproximable, there does not exist a class of policies for the pursuers which could approximate a provably ‘good’ solution. Therefore, we will define a policy for the pursuers that has some intuitive properties which we will later prove. Because the pursuers cannot communicate, we need to partition the road network into a set of induced subgraphs. Then, assign each pursuer to a sub-graph so that each pursuer can guarantee capture in their sub-graph. There are far too many ways one could choose to partition a graph. So, we will define the sub-graphs by first finding “Sweep-Covers” of the original graph. A sweep-cover is a collection of sets of sibling nodes in a tree that separates the source node from all the leaf nodes such that no two nodes in the sweep-cover are an ancestor or descendant of one another [15].

**Definition 2 (Sweep-Cover).** Let \( S \subset 2^{V_R} \) be a collection of non-empty sets of nodes where \( \forall S_i \in S, S_i \subset V_R \). Denote the sets \( A_S \) and \( D_S \) to be the set of all ancestors and the set of all descendants, respectively, of the nodes in \( \bigcup S_i \). We say \( S \) is a sweep-cover of \( \mathcal{R} \) if it fulfills the following conditions.

1) Each distinct pair \( S_i, S_j \in S \), with \( S_i \neq S_j \), is disjoint i.e., \( S_i \cap S_j = \emptyset \).
2) Each set may only contain sibling nodes, i.e. nodes which share a parent.
3) \( \bigcup_{S_i \in S} S_i \cup A_S \cup D_S = V_R \).
4) For any unique pair of nodes in \( S_i, v_i, v_j \in \bigcup S_i \), \( v_i \) cannot be a descendant or ancestor of \( v_j \).

**Definition 3 (Depth).** Let \( \mathcal{R} = \{V, E\} \) be a directed tree, with a root node, \( s \in V \). We define the depth of a given node \( v \in V \) to be the number of edges in the unique path from \( s \) to \( v \) in the tree. We assume that the depth of node \( s \) is 0.

**Definition 4 (Linear Path).** Let \( \mathcal{R} = \{V, E\} \) be a directed tree. We define a linear path to be an ordered set of nodes \( L_p = \{v_1, v_2, ..., v_m\} \) in \( \mathcal{R} \) such that \( \forall v_i \in L_p \setminus v_m : (v_i, v_{i+1}) \in E \land \nexists (v_i, v_{j \neq i+1}) = \emptyset \). In other words, a linear path is a chain of nodes in \( \mathcal{R} \) where each node from \( v_1 \) to \( v_{m-1} \) has only one outgoing edge to the next node in the chain.

**Definition 5.** Consider an instance of MPDP where the road network \( \mathcal{R} \) is a tree. For each node \( v \in V_R \), define \( \mathcal{L}(v) \) to be the descendant of \( v \) (or \( v \) itself) with the
largest depth such that if the evader passes through \( v \), it is also guaranteed to pass through \( E(v) \).

**Definition 6** (Induced Sub-Graphs from Sweep-Covers). Given a sweep-cover \( S \) from a tree \( \mathcal{R} = \{V, E\} \), for each \( S_i \in S \) construct a tree \( E_i = \{V_i, E_i\} \) in the following manner. Define \( V_i \) by taking every node which is an ancestor or descendant of the nodes in \( S_i \). Define \( E_i \) to be the set of all edges in \( E \) which can be constructed from the nodes in \( V_i \). We say the collection \( B = \{B_1, B_2, \ldots, B_i\} \) is a collection of induced sub-graphs defined by \( S \).

**Lemma 1.** Consider a collection of induced sub-graphs \( B \) defined by a sweep-cover \( S \) from a tree \( \mathcal{R} \) with root node \( s \in V \). Then, for each \( B_i \in B \) created from \( S_i \in S \) there exists a linear path in \( B_i \) from the root node of \( \mathcal{R} \) to the parent of the nodes in \( S_i \), denoted \( v_p \). Which further implies that \( v_p \) is the lowest known descendant of \( s \) if the evader is constrained to travel along the edges in \( B_i \).

**Proof.** By definition of a tree, there cannot exist multiple incoming edges to a node. Therefore, all nodes, with the root node, in the tree have exactly one parent. Then, all ancestors of \( v_p \) have exactly one parent except the root node. By definition of \( E_i \), for all ancestors of the nodes in \( S_i \), there does not exist edges to nodes outside of \( V_i \). Therefore, each node between the root node and \( v_p \) has one incoming edge from its parent and one outgoing edge to a child node which must also be an ancestor of \( v_p \). By definition, this is a linear path. Therefore, it is a linear path, the evader can only travel from \( s \) to \( v_p \) if it is constrained to \( E_i \). Therefore, \( v_p \) is the lowest known descendant of \( s \) on \( B_i \).

**Corollary 1.** Consider a collection of induced sub-graphs \( B \) created from a sweep-cover \( S \) defined on a tree \( \mathcal{R} \). Then, \( \bigcup_{B_i \in B} B_i = \mathcal{R} \).

**Proof.** By a simple extension of Definition 2.3, \( \bigcup_{B_i \in B} V_i = V \). \( \bigcup_{B_i \in B} E_i = E \) naturally follows by definition of \( E_i \).

**Corollary 2.** Consider an instance of MPDP where the road network \( \mathcal{R} \) is a tree. Given any sweep-cover \( S \) of \( \mathcal{R} \), the evader is guaranteed to pass through one node in \( S \).

**Proof.** We will prove this by contradiction. Suppose there exists a path in \( \mathcal{R} \) from the root node to a goal node which does not contain a node in \( S \). By definition of a tree, there cannot exist a node with more than one incoming edge. So, the final node in this path must be a goal node which is not a descendant of the nodes in \( S \) or a node in \( S \). However, this violates Definition 2.3 because a goal node cannot be an ancestor of any other node. Therefore, \( S \) cannot be a sweep-cover which is the contradiction.

**A. Multiple Pursuers**

**Definition 7.** (Pursuer Groups) Consider an instance of MPDP with a set of pursuers \( \mathcal{P} = \{p_1, p_2, \ldots, p_k\} \) of size \( k \). \( \forall t \in \mathbb{R}_{\geq 0} \), we say \( \mathcal{P}' \subseteq \mathcal{P} \) is a pursuer group at time \( t \) if \( \mathcal{H}_{p_1}(t) = \mathcal{H}_{p_2}(t) \ldots \mathcal{H}_{p_k}(t) \). Let \( \mathcal{P}_f(t) \subseteq 2^\mathcal{P} \) be the set of all pursuer groups at time \( t \). Upon visiting any node, a pursuer group may partition into new pursuer groups and each pursuer group will travel to a different node. We only define new pursuer groups based on partitioning of a previous pursuer group so the collection of all pursuer groups will always be a partition of the original pursuer group. Therefore, all properties of a partition apply to a collection of pursuer groups (disjointedness, etc.).

**Definition 8** (Group-Sweeping Policies). Let \( \Pi \) be the set of pursuer policies that satisfy the following conditions at all times \( t \geq 0 \): Consider the set of pursuer groups \( \mathcal{P}_f(t) \) and a partitioning of \( \mathcal{R} \) into a collection of sub-graphs \( \mathcal{R} \mathcal{P}_f(t) \) of size \( k \). \( \forall t \in \mathbb{R}_{\geq 0} \), we say \( \mathcal{P}' \subseteq \mathcal{P} \) is a pursuer group at time \( t \) if \( \mathcal{H}_{p_1}(t) = \mathcal{H}_{p_2}(t) \ldots \mathcal{H}_{p_k}(t) \). Let \( \mathcal{P}_f(t) \subseteq 2^\mathcal{P} \) be the set of all pursuer groups at time \( t \).

1. Define a sweep-cover \( S \) of size \( m \leq |\mathcal{P}_i| \) on the sub-tree rooted at \( u \).
2. Partition \( \mathcal{R}_j \) into a collection of \( m \) induced sub-graphs defined by \( S \) and denoted as \( B \).
3. Partition \( \mathcal{P}_i \) into a collection of \( m \) pursuer groups denoted as \( \mathcal{P}' \).
4. Define a bijection \( \mathcal{Q} \) from the set \( \mathcal{P}' \) to the set \( B \).
5. Remove \( \mathcal{R}_j \) from \( \mathcal{R}_f(t) \) and replace it with the induced sub-graphs in \( B \).
6. Remove \( \mathcal{P}_i \) from \( \mathcal{P}_f(t) \) and replace it with the pursuer groups in \( \mathcal{P}' \).
7. Remove the mapping from \( \mathcal{P}_i \) to \( \mathcal{R}_j \) in \( \mathcal{Q}(t) \) and replace it with all mappings in \( \mathcal{Q}'(t) \).
Note that by Corollary 1, any partitioning of \( R \) by the above method will guarantee that the union of all graphs in \( R_{ij}(t) \) is always the original graph. Therefore, if the each pursuer group can guarantee capture in its assigned sub-graph assuming the evader is constrained to its sub-graph, it can guarantee capture of the evader in the overall graph. Also, by definition of a tree a collection of induced sub-graphs from a sweep-cover cannot share descendants of the nodes in the sweep-cover which defined them. Otherwise, there exists a descendant which has two ancestors. Therefore, the evader cannot move between induced sub-graphs defined by sweep-covers. So, if a pursuer group has determined that the evader will never visit all the nodes at depth \( m - 1 \) in its sub-graph, it can assume that the evader will never visit the nodes at depth \( m \). Inductively, this means the pursuer can assume all nodes at depth greater than \( m - 1 \) are green. If a pursuer group has determined the evader is not in its assigned sub-graph, it may stop at any node.

**Definition 9 (Node-Sweep).** For all subsets nodes \( N \subseteq V_R \), a pursuer group is said to perform a node-sweep of \( N \) if it visits the nodes of \( N \) in some specified order to determine the state, green or red, of the nodes.

**Definition 10 (Group-Sweep).** Consider a pursuer group \( P' \in P_g(t) \) at some time \( t \). Suppose \( P' \) has determined that the evader is in a sub-tree \( T_u \) rooted by a node \( u \in V_R \). Consider some \( m \leq |P'| \) and define a partition \( P = \{P'_1, P'_2, ..., P'_m\} \) of \( P' \) into \( m \) sets such that each distinct set is disjoint and non-empty. Then, define a sweep-cover \( S = \{S_1, S_2, ..., S_m\} \) of size \( m \) and a bijection \( Q \) from the set of pursuer groups \( P \) to the collection \( S \). Each pursuer group must then perform a node-sweep on its assigned set of nodes to determine the state of each node. If a pursuer group finds a red node, the pursuer group will then perform a group-sweep on a sweep-cover of the tree rooted by the red node.

A group-sweep simplifies the partitioning operations of a group-sweeping policy and takes advantage of Lemma 1. By the triangle inequality, it is always best for a pursuer group to travel directly to one of the largest depth nodes allowed by a group-sweeping policy. By definition of a group-sweeping policy, each pursuer group can assume the evader is constrained to its sub-graph. Lemma 1 implies that if the evader is constrained to an induced sub-graph \( B_i \in B \) defined by the set \( S_i \subseteq S \), then it is guaranteed to travel to the parent of the nodes in \( S_i \), denoted \( v_p \). Assume that \( v_p \) is at depth \( m - 1 \). Then, the pursuer group knows the state of all nodes from depth 0 to \( m - 1 \) and it can visit the nodes in \( S_i \) immediately. By Corollary 2, it is guaranteed that the evader will pass through one of the nodes in \( S \). So, at least one pursuer group will be able to continue with a group-sweeping policy to capture the evader. However, it cannot know the states of the nodes in \( S \) until it visits them. Therefore, traveling directly to a node in \( S_i \) to determine the state of all nodes in \( S_i \) is optimal by the triangle inequality for a given sweep-cover and partition of \( P' \). So, a group-sweep is always optimal following the partitioning operation of a group-sweeping policy.

We will spend the remainder of the paper defining and proving calculations for finding the best possible group-sweeping policy for a given instance of MPDP by using group-sweeps.

![Figure 1](image_url)

Figure 1 provides an illustration of how a group-sweep works. Define \( T_u \) to be the sub-tree rooted at \( u \). Assume a pursuer group, \( P \), has determined that the evader is in \( T_u \) based on the states of \( v_u \) and \( v_e \), i.e. both are in the green state after the evader should have arrived. So, the pursuer group partitions into smaller pursuer groups where each travels to perform a node-sweep on its assigned set within the sweep-cover, \( S \), of \( T_u \). In this instance, \( S = \{\{v_{d_1}, v_{d_2}\}, \{v_{c_1}, v_{c_2}\}\} \) and node-sweeps are \( N'_1 = \{v_{d_2}, v_{d_1}\} \) and \( N'_2 = \{v_{c_2}, v_{c_3}\} \).

**B. Computing Maximum Pursuer Delay for an Instance of MPDP**

In this section we will provide methods to calculate MPDP using sweep-covers.

**Definition 11 (Evader Arrival Time at Node).** Given a tree network \( T = \{V_T, E_T\} \), for each node \( v \in V_R \), let \( t(v) \) be the evader distance from the source node \( s \) to node \( v \) (i.e., it is the time at which the evader would pass through node \( v \) if its path goes through that node).

**Definition 12 (Latest Pursuer Group Arrival Time at Node).** For each pair of nodes, \( v, u \in V_R \), let \( D(v, u, k) \) be the latest time at which a pursuer group of size \( k \) can arrive at node \( v \) and still be guaranteed to capture the
Lemma 2. For every goal node $u$, given that the evader’s path goes through node $u$.

Definition 13 (Latest Time to Begin Sweep at Node). Given a set of nodes from a sweep-cover $N \in S$, a starting node $w \in N$, and a pursuer group of size $k$, let $\Upsilon_N(w, k)$ be the latest time that the pursuer group can begin a node-sweep on $N$ starting from $w$ and still be guaranteed to eventually capture the evader, given that the evader is guaranteed to pass through a node in $N$. We define $\Upsilon_N(w, k) = -\infty$ if it is not possible to guarantee (eventual) capture via a node-sweep on $N$. If $N$ consists of a single node, $w$, then $\Upsilon_N(w, k) = D(w, w, k)$.

Definition 14 (Latest Time to Arrive at a Node and Sweep Other Nodes). We use $\Lambda_N(v, k)$ to denote the latest time that a pursuer group of size $k$ can arrive at node $v$ and still be guaranteed to capture the evader with a group-sweeping policy if the evader travels through any of the nodes in a set, $N$, of nodes from a sweep-cover.

Definition 15 (Feasible Node-Sweep). Consider a set of nodes from a sweep-cover, $N = \{v_1, v_2, \ldots, v_p\}$, an assigned pursuer group and some permutation of $N$ defined as the node sequence $N' = \{v_{n_1}, v_{n_2}, \ldots, v_{n_m}\}$. We say that $N'$ is a feasible node-sweep starting at $v_{n_1}$ if it is possible to guarantee (eventual) capture of the evader after following the node-sweep and then performing a group-sweep on a sweep-cover of any sub-tree rooted by any nodes in $N$.

C. Calculating Latest Pursuer Group Arrival Time at a Node

The following result immediately follows from the definitions of $D(v, u, k)$ and $t(v)$.

Lemma 2. For every goal node $v_g \in G$ paired with every sibling node, $v_s$ of $v_g$:

$$D(v_i, v_g, \cdot) = t(v_g) - t_p(v_i, v_g)$$

(1)

Proof. By definition, a pursuer must arrive at a goal node before or just as the evader reaches it for capture. Therefore, if a pursuer arrives at a sibling of $v_g$ later than $t(v_g) - t_p(v_i, v_g)$ the evader will have left the network by the time the pursuer arrives at $v_g$. This applies regardless of the pursuer group size. Note that for $v_i = v_g$, $D(v_g, v_g, \cdot) = t(v_g) - t_p(v_g, v_g) = t(v_g)$.

Lemma 3. Consider a pursuer group of size $k$ located at node $v$ that must perform a node-sweep on a set of nodes $N = \{v_1, v_2, \ldots, v_p\}$ from a sweep-cover. Let the node $v_a$ be the parent of all nodes in $N$. If there does not exist a feasible node-sweep of $N$ then $\Lambda_N(v, k) = t(v_a) - t_p(v_a)$. Otherwise,

$$\Lambda_N(v, k) = \max_{w \in N} \{\Upsilon_N(w, k) - t_p(v, w)\}$$

(2)

Proof. By definition, the latest time to begin a node-sweep with a pursuer group of size $k$ on the set $N$ starting at node $w$ is $\Upsilon_N(w, k)$. We find the maximum departure time to leave node $v$ and begin a node-sweep on $N$ by comparing all departure times to travel to every starting node, $w \in N$. However, if no node-sweep is feasible, then we can still guarantee capture by arriving at the parent of all nodes in $N$ at the same time as the evader, which is computed by $t(v_a) - t_p(v, v_a)$.

Definition 16 (Ordered Integer Partitions). We denote $\Gamma_n^k$ to be the collection of all ordered integer partitions of the integer $k$ where each partition contains $n$ parts. Specifically, an ordered integer partition of $k$ is an ordered set of $n$ elements where each element is an integer less than or equal to $k$. The sum of all elements for that partition is $k$. This is simply a set theoretic formalism of an ordered integer partition. I.e. for a given ordered integer partition of $k = k_1 + k_2 + \ldots + k_n$, the corresponding ordered partition set is $K = \{k_1, k_2, \ldots, k_n\}$

Lemma 4. Consider a pursuer group, $P_s \subseteq P$, of size $k$, located at node $v$. Suppose $P_s$ has determined that the evader must be in the tree rooted at a node $u$, denoted $T_u$. The maximum delay for which $P_s$ can guarantee capture of the evader by performing a group-sweep on $T_u$ is given by Algorithm 1.

Proof. By definition, we want to calculate the maximum delay a pursuer group of size $k$ can arrive at node $v$ and capture the evader by a group-sweep on a sweep-cover of the sub-tree rooted at $u$, i.e. $T_u$. Therefore, we consider all feasible sizes of sweep-covers (line 1) on $T_u$. The maximum possible size of a sweep-cover must be $k$ because if it were any larger then there would be a sweep-cover with at least one set without an assigned pursuer. One can see that lines 2 to 7 consider all possible splits of $k$, i.e. all ordered partitions of $k$, into all possible sweep-covers.

Consider a collection of pursuer groups defined as $P' = \{P_1, P_2, \ldots, P_p\}$ from a partition at node $v$ and a sweep-cover, $S'$, for $T_u$. For any pursuer group $P_j \in P'$ assigned to the set, $S_j \subseteq S'$, the maximum delay to begin a group-sweep is limited by the minimum delay for any pursuer group in $P'$ to leave and perform a node-sweep on $S_j$. Otherwise, at least one pursuer group arrives too late to guarantee capture. By definition, the maximum delay for any $P_j$ of size $|P_j|$ to leave node $v$ and perform a node-sweep on $S_j$ is $\Lambda_{S_j}(v, |P_j|)$. Therefore, the minimum across these times is the maximum possible delay for the group-sweep to begin on the particular
Algorithm 1 Calculate $D(v, u, k)$

**Input:** A road network with root node $u \in R$, denoted as $T_u$, an integer $k$ for the size of the pursuer group, and a node $v \in R$ where the pursuer group is currently located.

**Output:** The latest time at which a pursuer group of size $k$ can arrive at $v$ to perform a split and group-sweep on $T_u$ and still be guaranteed to capture the evader if the evader is in $T_u$.

1. for $n = 1$ to $k$ do
2. Let $S$ be all sweep covers of size $n$ of $T_u$.
3. for each sweep cover $S \in S$ do
4. for each ordered integer partition $K \in \Gamma_n^k$ (Definition 16) do
5. for $i = 1$ to $n$ do
6. Let $S_i \in S$ be the $i$th set of nodes in $S$
7. Let $k_i \in K$ be the $i$th integer in $K$
8. Calculate $\Lambda_{S_i}(v, k_i)$ according to Lemma 3
9. Return $\max_{n \in \{1, \ldots, k\}} \{\max_{S \in S} \{\min_{K \in \Gamma_n^k} \{\min_{i \in \{1, \ldots, n\}} \Lambda_{S_i}(v, k_i)\}\}\}$

**D. Calculating the Latest Time to Begin a Node Sweep at a Node**

The following result characterizes the feasibility of a node-sweep starting at a given node, in terms of the quantities $t(\cdot)$ and $D(\cdot, \cdot, k)$.

**Lemma 5** (Node-Sweep with Pursuer Group). Consider a set of nodes, $N$, and a node-sweep permutation of $N$ denoted as $N'$ with node sequence ${v_1, v_2, \ldots, v_{\mu-1}, v_\mu}$. Then we say that $N'$ is a feasible node-sweep if and only if the following conditions are met:

- A pursuer group of size $k \geq 1$ can follow the sequence $v_1, v_2, \ldots, v_{\mu-1}$ such that it visits each node $v_e$ in the interval $[t(v_e), D(v_e, v_e, k)]$, $e \in \{1, 2, \ldots, \mu - 1\}$.
- The pursuer group arrives at $v_{\mu-1}$ before or at $D(v_{\mu-1}, v_\mu, k)$.

**Proof.** A pursuer group must know the eventual state of every node that it visits in a node-sweep (Definition 9). Therefore, the pursuer group must visit each node $v \in N'$ after or at $t(v)$. Suppose some $v \in N'$ is found to be in the red state, then the pursuer group is required to guarantee capture by a group-sweeping policy starting at that particular node. Therefore, for all $v \in N'$, the pursuer group has to arrive at node $v$ before or at $D(v, v, k)$ but after $t(v)$, otherwise capture is not guaranteed by definition. Now, suppose that the first $\mu-1$ nodes are in a green state after the evader would have arrived at them. Then the pursuer group knows that $v_\mu$ is in the red state (assuming the evader is within this subset of the complete graph). In order to guarantee capture of the evader, the pursuer group must arrive at $v_{\mu-1}$ before or at $D(v_{\mu-1}, v_\mu, k)$. By definition, capture cannot be guaranteed any other way.

**Lemma 6.** Consider a set of nodes $N$. For any two nodes $v, u \in N$, $D(v, u, k) \geq t(u) - t_p(v, u)$. If $\Upsilon_v(k, S_i) \neq -\infty$, then $D(v, u, k) \geq \Upsilon_v(k, N) \geq t(v)$.

**Proof.** Capture is guaranteed if a pursuer group arrives at node $v$ at time $t(v)$, therefore $D(v, u, k) \geq t(v)$ and the same follows for any node in $N$. Accounting for the travel time from node $v$ to $u$ and the triangle inequality, it can be seen that $D(v, u, k) \geq D(u, u, k) - t_p(v, u) \implies D(v, u, k) \geq t(u) - t_p(v, u)$. Suppose that $\Upsilon_v(k, N) \neq -\infty$, which means that there is a time in which a pursuer group can begin a node-sweep at node $v$ and still be guaranteed to capture the evader via a group-sweeping policy. If the evader has gone through $u$ and $\Upsilon_v(k, N) > D(v, u, k)$, then this would contradict the definition of $D(v, u, k)$. Therefore, $\Upsilon_v(k, N) \leq D(v, u, k)$. By using the definition of a node-sweep, the pursuer group must determine the states of each node that it visits in $N$. Furthermore, the pursuer group must depart node $v$ no earlier than $t(v)$ (since that is when the evader would get to that node). Because the pursuer group is able to arrive and leave a node at the same time, the latest time to begin a node-sweep at node $v$ must satisfy $\Upsilon_v(k, N) \geq t(v)$, proving the claim.

**Lemma 7.** Consider a node-sweep $N' = \{v_1, v_2, \ldots, v_\mu\}$ where $\forall v_j \in N'$, $D(v_j, v_j, k) = t(v_j)$. Assume a pursuer group of size $k$ is assigned to sweep $N'$ and $N'$ is ordered such that $t(v_h) \leq t(v_{h+1})$ for all $1 \leq h \leq \mu - 1$. Then capture is guaranteed if and only
if the evader passes through the parent of all nodes in
$N'$, and
\[ \forall h \in \{1, \ldots, \mu - 1\} : t_p(v_h, v_{h+1}) \leq t(v_{h+1}) - t(v_h) \] (3)
If this condition is satisfied, then $\Upsilon_{v_1}(k, N) = D(v_1, v_1, k) = t(v_1)$. Otherwise, $\Upsilon_{v_1}(k, N) = -\infty$.

Proof. If the evader is attempting to travel to any node in $N$, then it must be captured by a pursuer group at
one of the nodes exactly as it arrives. For this to be possible, the pursuer group must visit each of the nodes
in $N$ in a sequence of increasing evader arrival times.
The group is required to leave each node no earlier than $t(w), \forall w \in N$. If the condition is met, a feasible node-
sweep exists equal to the earliest evader arrival time. If
this condition is not met, then either the pursuer group
has to leave a node before the evader arrives at that node
or the pursuer group does not have enough time to travel
to the next node. Then, capture is not possible. \qed

III. ALGORITHM TO CALCULATE MAXIMUM PURSUER DELAY

In this section we provide Algorithm 2 to calculate
the maximum pursuer delay given $k$ pursuers as well as
make observations on optimal policies which proves our
policy can guarantee an optimal maximum pursuer delay
time given enough pursuers.

A. An Algorithm to Calculate $D(v, u, k)$

Lemma 2 will provide all $D(v, u, k)$ for all goal nodes.
Then, we will use a dynamic programming approach to allocate nodes into induced sub-graphs based on the
results from Lemma 4. The final result is $D(v_1, v_1, k)$
which is the maximum pursuer delay for a pursuer group
of size $k$ to perform a group-sweep on the tree rooted
by $v_1$, i.e. $\mathcal{T}$.

The overall complexity of Algorithm 2 is $O(\Delta n k^k)$
where $\Delta$ is the maximum out degree of all nodes in
the road network, $n$ is the number of nodes in the road
network, and $k$ is the size of the initial pursuer group.

B. Optimal Results by Graph Constraints

In this subsection, we characterize the maximum pursuer
delay for an infinite number of pursuers under any
policy. We denote this to be the graph-constraint because
it is dependent on goal node placements and the structure
of the road network.

Corollary 3. Consider some DAG, $\mathcal{R}$, with root node
$s$ and a set of goal nodes, $\mathcal{G}$. If the size of the initial
pursuer group is $k$ and $k \geq |\mathcal{G}|$, then the optimal policy
is to partition the initial pursuer group into $|\mathcal{G}|$ pursuer
groups and assign each to travel directly to a different
goal node. By the triangle inequality, pursuer groups
lose maximum delay time if they travel to any other node
before a goal node and because the evader is guaranteed
to travel to one of the goal nodes, the pursuer groups
can travel directly to them.

Lemma 8. If $|\mathcal{G}|$ pursuer groups are assigned to $|\mathcal{G}|$
goal nodes, then the maximum pursuer delay is:
\[ \min_{\forall v \in \mathcal{G}} \{ t(v_g) - t_p(s, v_g) \} \] (4)

Proof. According to Corollary 3, the optimal policy
given a large enough initial pursuer group is for the split
pursuer groups to be assigned one goal node such that all
goal nodes have a pursuer group. The maximum delay
for each pursuer group is equal to $t(v_g) - t_p(s, v_g)$ where
$v_g$ is the assigned goal node. Therefore, the minimum
delay of all pursuer groups to arrive at their respective
goal nodes is the maximum time the pursuer groups can
leave the source node. Otherwise, at least one pursuer
group won’t be able to arrive at their goal node in
time.

Lemma 9. Given enough pursuers, Algorithm 2 is
guaranteed to find a policy which achieves the optimal
maximum pursuer delay.

Proof. Consider the set of all goal nodes in some DAG,
$\mathcal{R}$, to be $\mathcal{G}$. If we construct a collection, $S$, of
singleton where each goal node in $\mathcal{G}$ is a singleton in $S$ then by description this collection is a sweep-cover for $\mathcal{R}$.
Given k pursuers, where $k \geq |\mathcal{G}|$, our algorithm considers
a surjection from $k$ pursuers to the sweep-cover $S$ because
our algorithm considers all possible sweep-covers of size $1$ to $k$. By an application of Lemma 4 and Lemma 7, we
can see that calculating the delay is equivalent to Lemma
8. \qed

Corollary 4. By an extension of Lemma 2, there are
cases where $k \geq |\mathcal{G}|$ is not required to achieve the graph
constraint maximum.

Proof. Suppose the goal node which limits the graph
constraint is in a set, $S_i$, of other goal nodes in a sweep-
cover of the complete graph. Then, by Lemma 7 it is
possible for there to exist a feasible node-sweep of $S_i$
with a single pursuer. Therefore, even if the remaining
sets of the sweep-cover are singletons of goal nodes,
this particular set contains a node-sweep of multiple
goal nodes. By Lemma 3 this sweep-cover will have the
same delay as the graph-constraint delay. This means
it requires fewer than $|\mathcal{G}|$ pursuers to group-sweep this
sweep-cover and achieve the graph constraint result. \qed

It is important to note that MPDP considers a fixed
$k$ and the graph constraint results make no claim to
Algorithm 2 Find Maximum Pursuer Delay for Group-Sweeping Policies on Tree Networks

Input: A road network $R = (\mathcal{V}_R, \mathcal{E}_R)$ and the amount of pursuers $k$. The nodes are assumed to be sorted according to a topological ordering.

Output: Maximum pursuer delay time $\tau$.

1: For each $v \in \mathcal{V}_R$, calculate evader arrival time $t(v)$.
2: For each goal node $v_g \in \mathcal{G}$, calculate $D(v_i, v_g, \cdot)$ for each sibling $v_i$ of $v_g$, including $v_g$, using Lemma 2
3: Let $J$ be the maximum depth of the tree.
4: for $i$ from $J$ to 0 do
5:   for Each non-goal node $u$ at depth $i$ do
6:      for $h$ from 1 to $k$ do
7:         for Each node $v$ in the set of siblings of $u$, including $u$ do
8:            Calculate $D(v, u, h)$ according to Lemma 4
9:      Store the optimal sweep cover and group-sweep information from Lemma 4 for $D(v, u, h)$

solving MPDP for any fixed $k < |\mathcal{G}|$. Rather, it shows that instances of MPDP for $k \geq |\mathcal{G}|$ can be solved in linear time and is therefore in the complexity class $L$. We provide Algorithm ?? in the appendix to calculate a value of $k$ for which MPDP can be solved in linear time by constructing larger sized sweep-covers from an initial sweep-cover as necessary. The $k$ found in Algorithm ?? is lower bounded by the number of nodes which have at least one goal node child and upper bounded by $|\mathcal{G}|$. It is unknown whether this $k$ is the minimal $k$ for solving MPDP in linear time. We leave it as an open question for future research.

IV. RESULTS AND DISCUSSION

A. Data

We set up a MATLAB simulation on randomly generated road networks of various out-degrees. We allowed for up to a 400 node road network with no more than 8 outgoing edges from a single node. For each road network, we calculate the graph-constraint to compare the advantage of adding multiple pursuers. For the case of $k = 1$, Algorithm 2 simplifies to the single pursuer case and is in fact identical to the algorithm proposed in [8]. For $k > 1$, we found significant advantages for adding 1 or 2 pursuers on practical road networks (i.e. out degree of 1-3 with random lengths) but as $k$ increased the advantage decreased until the delay plateaued at the graph-constraint. While the algorithm cannot guarantee that we minimize the number of pursuers required to achieve the graph-constraint, it can guarantee an optimal delay given enough pursuers according to Lemma 9.

B. Discussion on Robustness to Failure

By restricting communication between pursuers, there is no way to adjust if a pursuer fails. Therefore, consider a variant of the pursuit evasion game where ground sensors have a new ability to transmit how many pursuers are currently at the sensor to all pursuers touching it. Then, if a pursuer in a pursuer group drops out mid flight, the remaining pursuers in the pursuer group will be aware at the next ground sensor. They can then consider a different group-sweep to the descendants of the ground sensor to accommodate for the dropped pursuer. This information is already calculated and stored in line 9 of Algorithm 2 so adding this fault tolerance will incur no space or time complexity penalties.

C. Conclusion

In this paper, we consider a problem where multiple pursuers enter a road network to capture an evader by obtaining information from ground sensors at intersections in the road network. First, we restricted the pursuers to a ‘group-sweeping’ policy in order to reduce the computational complexity in exchange for an approximate solution. Then, we provided an algorithm to calculate the maximum delay to send in the pursuers and still guarantee capture of the evader under our policy. Using this policy and the structure of the road networks, we proved that our algorithm will find the optimal maximum pursuer delay under any policy given a non-optimal amount of pursuers. This result is due to the fact that
the best possible maximum pursuer delay is constrained by goal node placements with respect to the source node even with an infinite number of pursuers.

Our algorithm is based on a new structure found in trees called Sweep-Covers which considers a coverage of the graph by descendant and ancestor relationships. These sweep-covers may become useful in future applications for drone flight when covering multiple ‘lanes’ of traffic in a road network. Future work can be done in finding tighter bounds for enumerating sweep-covers on infinite, labeled, and directed trees. There might be a connection between enumerating sweep-covers and Catalan numbers. In fact, the number of sweep-covers on an ILD tree of degree 2 reduces to a Catalan number. Also, with the simplicity that sweep-covers bring, it may be more feasible to consider other Pursuit Evasion scenarios such as multiple evaders.

REFERENCES