2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Asymptotically-Optimal Multi-Robot Visibility-Based Pursuit-Evasion

Nicholas M. Stiffler Jason M. O'Kane

***Abstract*— The multi-robot visibility-based pursuit-evasion problem tasks a team of robots with systematically searching an environment to detect (capture) an evader. Previous techniques to generate search strategies for the pursuit team have shown to be either computationally intractable or permit poor solution quality. This paper presents a novel asymptotically optimal algorithm for generating a joint motion strategy for the pursuers. To explore the space of possible pursuer motion strategies, the algorithm utilizes a trio of hierarchical graph data structures that each capture certain elements of the problem such as connectivity (valid single pursuer motion), coordination (multiple pursuer motion), and tracking information (evaluating where an evader may be). The algorithm is inspired by well-known methods in the motion planning literature and inherits its asymptotic optimality from those techniques. In addition, we describe a method that can improve upon solutions found during the formative stages of the main algorithm, using a "fast-forward" approach that foregoes guarantees of asymptotic optimality, implementing heuristics that concentrate future samples into improving the path quality of the nominal solution. The algorithms were validated in simulation and results are provided.**

I. Introduction

Scenarios where an agent, or group of agents, attempts to track down and locate members of another group form the foundation of the problems known as *pursuit-evasion* problems. These problems are often formulated as a two-player game to capture the competing interests of the two groups. The searchers, generally referred to as *pursuers* attempt to detect or capture the members of the other group, the *evaders*. In many instances, the evaders may not necessarily be actively evading the pursuers. However, the game formulation is beneficial as it leads to "winning conditions" for the players which are useful in evaluating the ability of the players to achieve their goal, namely can the pursuers guarantee capture/detection of their target. In fact, a plethora of everyday tasks can be constructed as pursuit-evasion problems such as environmental surveying [4], [20], [43], search-and-rescue [3], [30], [31], and monitoring/surveillance [2], [13], [29].

This paper considers a geometric form of the pursuit-evasion problem where a group of pursuers are tasked with exploring a polygonal environment to determine the presence or absence of an intruder (the evader). This exploration presents a greater computational challenge compared to other tasks in the literature, like coverage and navigation, because it involves the additional requirement of finding the evader, who is capable of moving within the environment. It is insufficient for the pursuers to "visit" every region in the environment. Rather, the pursuers must reason over the space of potential evader motions. Specifically, the pursuers must reason about the areas of the environment currently beyond the view of their sensing capability and how these regions evolve as the pursuers move within the environment.

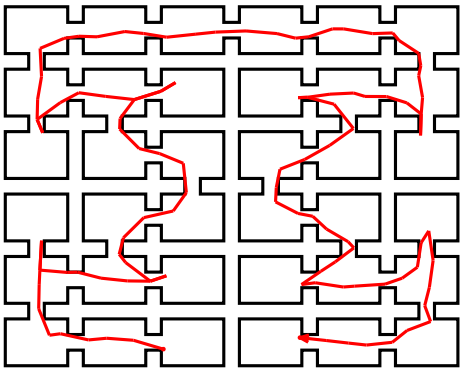

Fig. 1. The "pure" algorithm was able generate a solution strategy for the 36-room environment shown above. The search produced a G_0 graph containing 1,019 nodes, a G_1 graph with 792 nodes, and a G_2 graph with 203,996 nodes.

An emerging trend is the study of multi-pursuer variants of the pursuit-evasion problem. The increased complexity that arises when incorporating multiple pursuers has historically been an inhibiting factor to investigating the multi-pursuer visibility-based pursuit-evasion problem [39]. However, a number of recent results have begun to explore the viability of applying common sampling techniques found in the motion planning literature to the pursuit-evasion domain [33], [40]. An important distinction between the two approaches is that, much like the difference between coverage/navigation and pursuit-evasion, the space over which we are searching is more complex than the standard configuration space in which many of these applications have shown utility. Thus, while existing techniques have been able to leverage sampling techniques to show tractability, they often emit solutions of poor quality.

This paper bridges that knowledge gap by introducing an asymptotically-optimal algorithm for computing solution strategies for the multi-robot visibility-based pursuit-evasion problem. Our specific contributions are:

1) An introduction of a hierarchical three-tier graph structure that maintains a separation of concerns when reasoning over the space of possible pursuer strategies.
2) An algorithm that uses the aforementioned data structure to generate asymptotically-optimal pursuit strategies.

N. M. Stiffler is with the Department of Computer Science at the University of Dayton, Dayton, Ohio, USA. J. M. O'Kane is with the Department of Computer Science and Engineering at Texas A&M University, College Station, Texas, USA. nstiffler1@udayton.edu, jokane@tamu.edu

3) A heuristic extension to this algorithm that takes formative solutions born of the asymptotically-optimal algorithm and does some targeted sampling to improve the solution quality of the reported solution more rapidly than the base algorithm.

The remainder of the paper is structured as follows: A review of related work (Section II) is followed by a precise statement of the problem (Section III). Section IV describes the three data structures leveraged by the algorithms. The primary asymptotically-optimal algorithm appears in Section V before discussing an adaptive heuristic-based variant (Section VI). Finally, Section VII presents a quantitative evaluation of the algorithms, just before we conclude in Section VIII.

II. Related Work

This work is at the intersection of search and target-tracking and sampling-based motion planning. As such, there are elements from both communities with whom this work draws inspiration and shares similarity.

A. Search and Target Tracking

Pursuit-evasion represents a specific subproblem within the broader field of search and target tracking. Born of the seminal work of Isaacs [19] and Ho et. al. [18] pursuit-evasion was first discussed in the context of game theory as a differential game [7], [9]. Parsons [34] and Petrov [35], [36] are generally acknowledged as being at the forefront of graph-based pursuit-evasion where the domain is modeled as a graph [1], [8], [34]. The visibility-based pursuit-evasion problem discussed in this paper has its roots in the work of Suzuki and Yamashita [42], which offered the first geometric formulation of the pursuit-evasion problem. Guibas, Latombe, LaValle, Lin, and Motwani's pioneering work [17] resulted in a complete algorithm for the single-pursuer visibility-based pursuit-evasion problem, which has since been iterated upon in a number of contexts with various constraints on the motion and sensing capabilities of the pursuer [6], [14], [27], [44]. By comparison, the multi-robot visibility-based pursuit-evasion problem is in its infancy [12], [16], [24].

B. Sampling-Based Motion Planning

Sampling-based motion planning is a subfield born out of the computational intractability of reasoning over vast search spaces. Two popular techniques, which originated as path planning algorithms for individual robots, are the Probabilistic Roadmap (PRM) [23] and Rapidly-Exploring Random Trees (RRTs) [25]. These methods have inspired a wide range of variants and generalizations [21], [45]. Karaman and Frazzoli [22] improved upon these techniques by providing asymptotically-optimal variants of the original algorithms.

One caveat of the aforementioned sampling-based methods is that they quite often suffer from the *curse of dimensionality* [5] whereby, as the number of dimensions increases, the number of samples required for adequate coverage of the space increases exponentially. It is non-trivial to adapt sampling-based methods to the multi-robot domain [37]. Intuitively, the number of samples required to adequately sample an exponentially larger underlying space makes the problem computationally challenging. Thus, we are left with a few techniques such as implicit representation [38], sparse well-represented samples representative of a larger space [28], and cleverly designed sampling to avoid scenarios where the local planning step is efficient but the connection problem is difficult [15], [46].[1]

III. Problem Statement

To formalize the multi-robot visibility-based pursuit-evasion problem considered in this paper, we begin by describing the models used to represent the environment, evader(s), and pursuit team. We then provide the formalism used by the pursuit team to reason about the observed and unobserved areas of the environment. We characterize solutions, which are the joint paths followed by our pursuit team, that ultimately lead to the determination of whether the evader(s) have been detected.

A. Environment, Evader, and Pursuers

The *environment* W is a closed, bounded, and connected polygonal region in $\mathbb{R}^2$. A single *evader* seeks to avoid detection by the pursuers by moving continuously within the environment with a finite unbounded speeds. We denote the evader's location, as a function of time, by the continuous function $e(t) : [0, \infty) \rightarrow W$, unknown to the pursuers. Observe that, because we plan for the worst case, any pursuit strategy employed by the pursuers that guarantees detection of a single evader can also guarantee detection for arbitrarily many evaders.

The motion of a single pursuer p can be described by a continuous function $p : [0, \infty) \rightarrow W$, so that $p(t) \in W$ denotes the position of the pursuer at time $t \geq 0$. The function p is called a *motion strategy* for the pursuer. This can be extended to a team of n pursuers, $\widehat{p} = (p_1, p_2, \ldots, p_n) : [0, \infty) \rightarrow W^n$, to discuss the *joint motion strategy* of the pursuers. Thus the n-robot *joint pursuer configuration* (henceforth referred to as JPC) at time t is defined as $\widehat{p}(t) = (p_1(t), p_2(t), \ldots, p_n(t)) \in W^n$.

The pursuers are each equipped with a sensor capable of detecting the evader. The sensor model assumes an omnidirectional line-of-sight with an unlimited range that extends unobstructed to the environment boundary. That is, a pursuer at point $q \in W$ can detect anything within its *visibility polygon* $V(q) = \{r \in W \mid \overline{qr} \subset W\}$. By extension, a team of pursuers creates a *visibility region* defined as the union of each pursuer's visibility polygon. The visibility region at time t is defined as $V(\widehat{p}(t)) = V(p_1(t)) \cup V(p_2(t)) \cup \cdots \cup V(p_n(t))$.

The time of capture for an evader following trajectory e and a team of pursuers executing joint motion strategy $\widehat{p}$ is defined as $t_c(\widehat{p}, e) = \min \{t \geq 0 \mid e(t) \in V(\widehat{p}(t))\}$.

[1] This observation appeared in [25] and is exactly the problem we face in the pursuit-evasion domain.

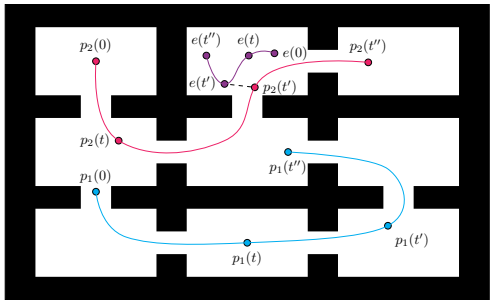

Fig. 2. Two pursuers p_1 and p_2 execute a joint motion strategy over the interval $[0, t'']$ where $0 < t < t' < t''$. Note that at time t' the pursuer p_2 is capable of detecting the evader e.

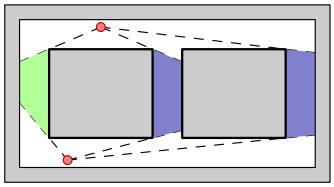

Fig. 3. A problem state with a JPC comprised of two robots (●), and a shadow label of 011 corresponding to a cleared shadow (■) and two contaminated shadows (■). This is one of eight possible problem states for the pursuers at this JPC.

The pursuers' objective is to establish visibility with the evader regardless of the evader's motion. A joint pursuer motion strategy $\widehat{p}$ is a *solution strategy* if there exists a finite time of capture, denoted $t_c(\widehat{p})$ and defined as $t_c(\widehat{p}) = \max_e \ t_c(\widehat{p}, e)$. Note that $t_c(\widehat{p})$ is the least upper bound over all valid evader motions when the pursuers employ joint motion strategy $\widehat{p}$. An optimal solution strategy occurs when $t_c(\widehat{p})$ is minimized. Note that there can be uncountably many such $\widehat{p}$ that result in $t_c(\widehat{p})$ being an infimum. Figure 2 illustrates the notation.

B. Shadows

We define the *shadow region*, $S(t) = W \setminus V(\widehat{p}(t))$ as the portion of the environment unobservable by any pursuer at time t. The maximal connected components of $S(t)$ are called *shadows*.

When reasoning about shadows, the pertinent piece of information is whether it is possible, based upon the pursuers' motions up to the current time t, for an evader to be concealed somewhere within the shadow. A shadow is deemed to be *cleared* if, based upon the pursuers' motions up to the current time t, it is guaranteed to not contain an unseen evader. Any shadow that is not cleared is called *contaminated*, and *may* still contain an evader within.

We employ a simple binary representation to describe the cleared/contaminated status for a shadow where a 1 indicates that the shadow is contaminated and a 0 indicates that the shadow is cleared. We can thus capture the status of multiple shadows by establishing an arbitrary but fixed ordering and concatenating the individual shadow statuses to create a binary string we refer to as a *shadow label*. As the pursuers move within the environment, the shapes of the shadows will change continuously, but the number of shadows and the cleared/contaminated status of individual shadows will change at discrete points in time.

C. Redefining the objective

A *problem state* is defined as the 2-tuple whose elements are a JPC and a shadow label. This representation is expressive enough to encode any information about past movements relevant to the problem. For a JPC with m shadows, there exist 2^m distinct problem states, each with a distinct binary shadow label. Figure 3 illustrates this idea.

As the pursuers conduct their search, the problem state is continually evolving. This affords the opportunity to state the pursuers' objective in terms of problem states as opposed to reasoning explicitly over the space of all possible evader trajectories. Under this viewpoint, a *solution strategy* is a motion strategy for the pursuers that, starting with a shadow label that is fully contaminated $(1 \cdots 1)$, forms a sequence of problem states ending in a problem state with a shadow label that is fully cleared $(0 \cdots 0)$. An optimal strategy is the shortest path that achieves this, measured by total travel for all of the pursuers.

IV. Data Structures: Environment Graph, JPC Graph, and Problem State Graph

This section introduces the data structures used to solve the multi-robot visibility-based pursuit-evasion problem introduced above. These data structures form the basis for the algorithms introduced in Sections V and VI. The main idea is to explore the space of possible solution strategies by forming three distinct graphs, each of which captures certain elements of the problem.

At the bottom level, the *environment graph* G_0 captures the connectivity of collision-free movements in the environment for a single pursuer. Next, the *JPC graph* G_1 describes coordinated motions that the entire team can make within the environment. Finally, within the *problem state graph* G_2, paths correspond to joint pursuer strategies, tracking the appropriate shadow labels at each step. These three graphs are constructed in harmony with one another, so that vertices in G_2 refer to G_1 and vertices in G_1 refer to G_0; details about these connections are described below. This "stack of graphs" structure enables the full complexity of the problem to be represented, while avoiding duplication of computation and enabling an effective balance of exploration of the space. The remainder of this section describes each of the graph data structures, including the operations that can be performed on each, in more detail.

A. The Environment Graph

The environment graph, denoted $G_0 = (V_0, E_0)$, is a weighted undirected graph. Each vertex in V_0 corresponds to a position within W. Each edge in E_0 corresponds to a line segment fully within W between the two vertices connected by the edge. An edge's weight is assigned according to the Euclidean distance between the endpoints.

a) Operation 1: Add and connect a specified point to G_0: The simplest operation in G_0 is to add a new vertex at a specified location $q \in W$. A new vertex, a_0, at this location is inserted into the graph. Then, the algorithm identifies the other vertices within some maximum connection distance

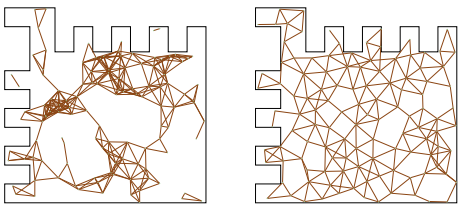

Fig. 4. [left] An example of graph G_0, constructed by repeated application of Operation 1 to insert random samples. [right] A different G_0, constructed by repeated application of Operation 2 with $\delta_0 = 0.99$ and $k_0 = 10$.

$d_{\max}$. For each such vertex $b_0 \in V_0$, the algorithm checks whether $\overline{a_0 b_0} \subset W$; if so, the algorithm inserts an edge between a_0 and b_0 into E_0. (The astute reader will note strong similarities between this operation and the standard probabilistic roadmap algorithm [23]; additional features, specific to visibility and pursuit-evasion, appear in G_1 and G_2 below.)

b) Operation 2: Add and connect a well-spaced point to G_0: Our algorithms also utilize a variant of Operation 1 that aims to select points for addition to the graph that are well-spaced throughout the environment. The objective is to ensure that the dispersion of the points in G_0 remains low, avoiding the clusters and barren spots that are characteristic of uniform random samples.

The value of low-dispersion sample sequences in general motion planning contexts has been known for some time [26]. Such factors are likely even more important in the context of these pursuit-evasion problems: Because so much additional processing is needed at the G_1 and G_2 levels, a small investment of time in constructing G_0 well can accelerate the algorithm as a whole.

To achieve this well-spaced sampling in a straightforward manner, the G_0 data structure maintains a real number χ_0 representing the minimum allowable distance between a potential new vertex and the nearest existing vertex. At the start, χ_0 is initialized to the diameter of W. Operation 2 uses χ_0 for rejection sampling: It generates a random sample $q \in W$ and performs a nearest-neighbor query to identify the closest existing vertex $q' \in E_0$. If the Euclidean distance between q and q' is less than χ_0, i.e $||q - q'|| < \chi_0$, the sample q is rejected and a new sample is selected. If k_0 samples in a row are rejected in this way — here k_0 is a positive integer parameter of the algorithm — then χ_0 is multiplied by a small constant factor $\delta_0 < 1$. When a sample q is ultimately found that is distance at least χ_0 from the existing vertices, it is added to the graph using Operation 1. Note that the rejection process cannot continue indefinitely, since repeated failures drive χ_0 exponentially toward 0. As illustrated in Figure 4, this process forces vertices added to G_0 to be well-spaced throughout W. In addition, the variable χ_0 can be used as a proxy for the dispersion of the samples represented by G_0.

B. The JPC Graph

The JPC graph, denoted $G_1 = (V_1, E_1)$, is also a weighted undirected graph. Each vertex in V_1 is a JPC; each edge in E_1 represents a direct collision-free path in W^n.

The graph G_1 builds upon G_0 by enforcing the following constraint: For every vertex $a_1 \in V_1$, consider the associated JPC $(p_1^{a_1}, p_2^{a_1}, \ldots p_n^{a_1})$. We require that every pursuer position $p_i^{a_1}$ in the JPC appears as a vertex in G_0. In practice, this is most easily implemented by constructing a_1 as a collection of n references to vertices in G_0. The advantage of this linking between G_1 and G_0 is that edges and vertices can be added to G_1 without any geometric computation to detect collisions, since the necessary information about collisions with the environment is fully encapsulated within G_0. This sort of construction shares some similarity with the dRRT* work of Dobson, Solovey, Shome, Halperin, and Bekris [11], but requires an explicit construction of the multi-robot graph because of the need, in this pursuit-evasion context, to track shadow labels.

In addition to tracking the structure of the graph G_1, this data structure also labels each edge $e \in E_1$ from $a_1 \in V_1$ to $b_1 \in V_1$ with a *shadow influence relation* R_e. This is a binary relation between the shadows at a_1 and the shadows at b_1. Specifically, for a shadow s_{a_1} at a_1 and another shadow s_{b_1} at b_1, if $(s_{a_1}, s_{b_1}) \in R_e$, then there exists a path segment for the evader starting in s_{a_1} and ending in s_{b_1} which remains unseen by the pursuers as they move from a_1 to b_1. This information is relevant for computing problem states: After a motion along e from a_1 to b_1, a given shadow s_{b_1} is contaminated if and only if there exists a contaminated shadow s_{a_1} for which $(s_{a_1}, s_{b_1}) \in R_e$. The shadow influence relation is computed each time an edge is added to E_1, and used in the construction of G_2, as described later. These shadow influence relations generalize the shadow influence caches recently introduced in a somewhat different context [32].

To construct G_1, we utilize two main operations that echo those for G_0.

a) Operation 3: Add and connect a specified JPC to G_1: A new vertex a_1 is added to V_1, at a JPC specified by indicating vertices in V_0 for each of the n pursuers. Then new edges in E_1 connect a_1 to any other vertex $b_1 \in V_1$ for which edges exist in E_0 to confirm that such motion is collision-free.

b) Operation 4: Add and connect a well-spaced JPC to G_1: Similar to Operation 2, we use rejection sampling to select JPCs, selected at random from $(V_1)^n$, maintaining a minimum distance from each other, with the minimum distance adjusted downward upon repeated failures. This operation is governed by parameters δ_1 and k_1, analogous to the δ_0 and k_0 utilized in Operation 2. When a suitably-spaced sample JPC is identified, we add it to G_1 using Operation 3.

C. The Problem State Graph

The final layer in our stack of graphs is the *problem state graph*, a directed graph $G_2 = (V_2, E_2)$. Each vertex in V_2 represents a full problem state; each edge in E_2 represents a feasible direct transition from one problem state to another. Analogously to the connection between G_0 and G_1, the problem state for each vertex in G_2 shares its JPC with some vertex in G_1. This constraint enables us to use the shadow

influence relations stored in G_1 to compute transitions in G_2 rapidly. Specifically, for a vertex $a_2 \in V_2$, we identify its associated $a_1 \in V_1$. Then each edge $a_1 \to b_1 \in E_1$ forms an out edge in G_2 from a_2.

Additionally, at each vertex $a_2 \in V_2$, the data structure also maintains two additional attributes: (i) A *parent* $\rho(a_2) \in V_2 \cup \{\text{NIL}\}$, either a reference to another vertex in V_2 or NIL. (ii) A *distance* $d(a_2) \in \mathbb{R}^{\geq 0} \cup \{\infty\}$, a nonnegative real number or infinity. These attributes play similar roles to the parent and distance attributes used in most implementations of Dijkstra's algorithm [10]: The parent $\rho(a_2)$ tracks the next vertex to visit along the best known path from a_2 to an all-clear vertex, and the distance $d(a_2)$ tracks the length of that path. Each new vertex a_2 begins with $\rho(a_2) = \text{NIL}$ and $d(a_2) = \infty$ if the problem state of a_2 has at least one contaminated shadow, or $d(a_2) = 0$ if the problem state of a_2 has all cleared shadows.

a) Operation 5: Add and connect vertices for all problem states at a given JPC: Given a specific JPC corresponding to a G_1 vertex $a_1 \in V$, this operation adds a collection of vertices to G_2 that share that same JPC, but differ in their respective shadow labels. Let k denote the number of shadows at the JPC in question. We create 2^k vertices, encompassing all possible shadow labels. Out edges from these new vertices can be constructed in a straightforward way from the corresponding edges in E_1: The targets of those edges indicate the correct JPC and the edge's associated shadow influence relation, applied to the k shadows at a_1, as label in the problem state of a_2, directly yield the resulting shadow labels.

b) Operation 6: Update parents and distances: The final basic operation is to inspect the parent and distance pointers in G_2 to determine whether additions to the graph since the previous call to this operation has created any shorter paths, and if so to update those attributes. This relies on the standard edge relaxation, wherein for any edge $e \in E_2$ connecting a_2 to b_2, if $d(a_2) > w(e) + d(b_2)$, we set $\rho(a_2) \leftarrow b_2$ and $d(a_2) \leftarrow w(e) + d(b_2)$. A call to Operation 6 performs these updates until $d(a_2) \leq w(e) + d(b_2)$ for every edge $e \in E_2$.

V. Algorithm: Asymptotically-Optimal

The three graph data structures G_0, G_1, and G_2 introduced in Section IV form the basis for the main contribution of this paper: an asymptotically optimal algorithm that solves the multi-robot visibility-based pursuit-evasion problem. The intuition is that as the number of samples approaches infinity, the solution provided by the algorithm will converge to the optimal solution.

The main idea, starting from empty graphs for G_0, G_1, and G_2, is to incrementally add to these graphs using Operations 2, 4, and 5 while maintaining the shortest path information in G_2 using Operation 6. (Operations 1 and 3 are used indirectly; recall that they are subroutines to Operations 2 and 4 respectively.)

The algorithm starts with a short initial phase of singular focus on G_0, ending when G_0 becomes a non-trivial (i.e. more than one vertex) connected graph. After that, the challenge in the algorithm is to determine where computational effort should be invested as time progresses. For simplicity, our algorithm always applies Operations 4, 5, 6 in tandem, so that G_2 always has vertices linked to each of the vertices in G_1 and the shortest path information in G_2 is always up-to-date. Thus, at each iteration of the main algorithm, it remains to determine whether to expand G_0, exploring the space of individual pursuer movements more fully; or to expand G_1 and G_2, exploring the space of coordinated movements and their impact on the shadow labels.

We resolve this tension using the minimum spacing variables χ_0 and χ_1. The intuition is to treat these values as proxies for the dispersion of the samples added thus far at each level. We aim to split the exploration evenly between these layers, but this is complicated by the fact while G_0 lives within W, G_1 lives in W^n, and therefore distances in G_1 have the capacity to be larger than in G_0. Consequently, instead of comparing χ_0 and χ_1 directly, we scale these values based on the maximum possible distance. Specifically, at each iteration, if $\chi_0 \leq \chi_1/\sqrt{n}$, we perform Operation 2 to add to G_0. Otherwise, we perform Operations 4, 5, and 6 to add to G_1. This process continues as long as time permits. At any point, if there exists a vertex $v_2 \in V_2$ whose problem state shows all shadows contaminated and for which $\rho(v_2) \neq \text{NIL}$, then a correct solution strategy starting from v_2 can be extracted from the graph. As the algorithm proceeds shorter solution paths may be found, gradually converging to the optimal solution.

Though space limitations prohibit a detailed argument for the asymptotic optimality of this approach, the intuition follows directly from the well-known arguments for asymptotic optimality of motion planners like RRT* and PRM*: For a given optimal solution, the 'rewiring' process ensures that the paths encoded in G_2 represent the shortest paths to all clear within that graph, and the underlying sampling process ensures that samples adequate to yield an close approximation of the optimal solution will eventually be included in G_2.

VI. Algorithm: Heuristic-based

Next, we describe a variation of the main algorithm introduced in Section V. The idea is to rapidly improve upon the correct but inefficient solutions found during the early stages of the search by targeting graph additions in places intuitively likely to lead to improvements.

To avoid the heavy computational burden that arises as the graphs grow in size, we begin by "*resetting*" the graphs G_0, G_1, and G_2 and seeding them with the points, JPCs, and problem states of the solution via repeated application of Operations 1, 3, 5. Using these newly formed sparse graphs as a basis, we employ a series of "fast-forward" operators designed to add specific vertices around the existing solution. The algorithm applies these operations repeatedly, selecting at random from a pool of operators in each of its iterations.

Several of these operators have proven effective in practice. Each one takes as input a solution strategy in the form

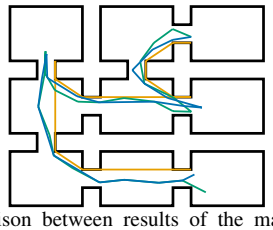

Fig. 5. A comparison between results of the main algorithm (■), the fast-forward solution (■), and a known optimal solution (■). Notice several improvements in the fast-forward solution, including the initial point (bottom right) being adjusted toward the optimal starting position and the motion along the middle corridor being been smoothed.

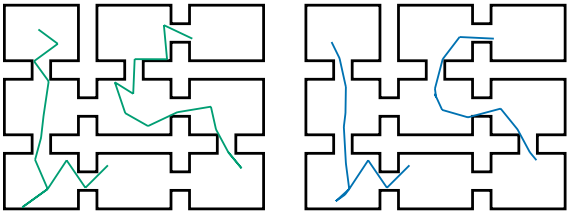

Fig. 6. Solution strategies generated by our algorithms. [left ■] The asymptotically optimal algorithm. [right ■] fast-forward algorithm. (The optimal solution is not shown; to the authors' knowledge, no existing algorithm is able to generate those path.)

of a sequence of JPCs and add new vertices to any of the three graphs with appropriate applications of Operations 1, 3, 5 before updating the distance calculations with Operation 6.

- *Short-cutting:* Select an arbitrary index i from the solution sequence and add a new JPC at the midpoint of the JPCs at indices $i-1$ and $i+1$ JPC.
- *Endpoint contraction:* Generate a new JPC between the first and second (or next-to-last and last) JPC of the solution sequence.
- *Guarding:* Select an arbitrary index from the solution sequence and generate a new JPC that holds one of the pursuers back at the sample position at that index.
- *Endpoint adjustment:* Generate a new JPC where each pursuer is within $d_{\max}$ of its position in the initial (or final) JPC.
- *Interior adjustment:* Select an arbitrary index from the solution sequence and generate a new JPC where each pursuer remains within $d_{\max}$ of its position in the original JPC.

Note that because these operations bias the selection of samples which can be added in future iterations, the usual argument for asymptotic optimality does not apply to these operations in isolation.

VII. Quantitative Results

This section evaluates the algorithms described in Sections V and VI using a implementation in Python 3 and executed on a computer with an Intel Core i7-10510U CPU with 16GB of RAM. All results in this section used $\delta_1 = \delta_2 = 0.99$, and $k_0 = k_1 = 10$. The implementation of the fast-forward-algorithm samples uniformly from the fast-forward operators described in Section VI.

Figures 1, 5, and 6 show solution strategies computed by this implementation. In particular, Figure 5 shows a comparison to the known optimal single-pursuer solution [41]. The solutions generated by the asymptotically optimal algorithm,

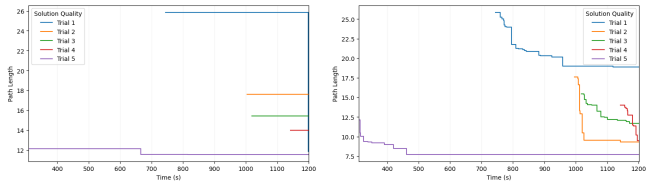

Fig. 7. Five trials illustrating the performance of the main algorithm [left] vs. the fast-forward algorithm [right] in the environment shown in Figure 3.

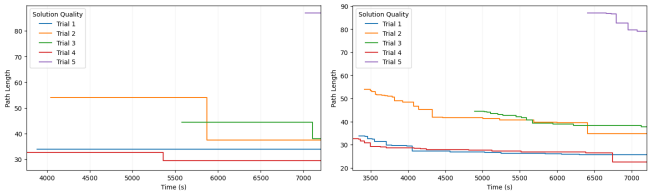

Fig. 8. Five trials illustrating the performance of the main algorithm [left] vs. the fast-forward algorithm [right] in the environment shown in Figure 6.

even in a relatively short execution, generally trace the same path as the true optimal solution; the fast-forward algorithm does so even more closely. Interestingly, in Figure 1, although G_0 and G_1 remain relatively small (approximately 1000 and 800 vertices, respectively), the G_2 graph already has more than 200,000 nodes. This is a direct consequence of the geometry of that environment, in which many JPCs have many disjoint shadows. Our implementation was unable to solve the two-pursuer instance of this problem without exhausting memory, because the larger G_1 led to excessive growth of G_2, far beyond the 200,000 vertices in Figure 1.

To measure the utility of fast-forward operators introduced in Section VI, we applied both the main algorithm and the fast-forward algorithm to the environments in Figures 3 and 6, conducting 5 trials. In each trial, both algorithms received the same seed for their pseudorandom number generators, so their constructions of all three graphs were identical until such time as the first solution was generated. For each trial, we measured the length of the best known solution, as a function of time throughout the algorithm's execution. Results appear in Figures 7 and 8.

Notice that, in the main algorithm, progress generally stalled after finding the first solution. In contrast, the fast-forward algorithm made rapid improvements to the initial solution in all cases. We conclude that, while it is possible for the main algorithm to generate solutions of high quality (See, for example, Trial 5 of Figure 7), a much more likely outcome is that the main algorithm will discover a solution strategy with ample opportunity for the style of local optimizations performed in the fast-forward algorithm.

VIII. Conclusion

This paper presented an asymptotically optimal algorithm for generating motion strategies for visibility-based multi-robot pursuit-evasion. Opportunities for additional research include reducing the number of parameters, adjusting the connection distance in G_0 as the algorithm proceeds (as in RRT*) and a more thorough exploration of the best ways to fuse the main 'pure' algorithm with the heuristic operators and the graph reset operation, to obtain fast performance while retaining the optimality guarantees.

REFERENCES

[1] T. V. Abramovskaya and N. N. Petrov, "The theory of guaranteed search on graphs," *Vestnik St. Petersburg University*, vol. 46, no. 2, pp. 49–75, 2013.

[2] T. Alam, M. M. Rahman, L. Bobadilla, and B. Rapp, "Multi-vehicle patrolling with limited visibility and communication constraints," in *Military Communications Conference*, 2017, pp. 465–479.

[3] R. Arnold, H. Yamaguchi, and T. Tanaka, "Search and rescue with autonomous flying robots through behavior-based cooperative intelligence," *Journal of International Humanitarian Action*, vol. 3, 12 2018.

[4] E. Aucone, S. Kirchgeorg, A. Valentini, L. Pellissier, K. Deiner, and S. Mintchev, "Drone-assisted collection of environmental dna from tree branches for biodiversity monitoring," *Science Robotics*, vol. 8, no. 74, p. eadd5762, 2023. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.add5762

[5] R. Bellman, *Dynamic Programming*. Princeton, NJ: Princeton University Press, 1957.

[6] D. Bhadauria, K. Klein, V. Isler, and S. Suri, "Capturing an evader in polygonal environments with obstacles: The full visibility case," *International Journal of Robotics Research*, vol. 31, pp. 1176–1189, 2012.

[7] S. Bhattacharya and S. Hutchinson, "On the existence of nash equilibrium for a two-player pursuit—evasion game with visibility constraints," *International Journal of Robotics Research*, vol. 29, no. 7, pp. 831–839, 2010. [Online]. Available: https://doi.org/10.1177/0278364909354628

[8] R. Borie, S. Koenig, and C. Tovey, "Pursuit-evasion problems," in *Handbook of Graph Theory*, J. Gross, J. Yellen, and P. Zhang, Eds. Chapman and Hall, 2013, ch. 9.5, pp. 1145–1165.

[9] D. Cardona, I. Becerra, and R. Murrieta-Cid, "On the equivalence of pursuer strategies and the lack of nash equilibrium in a visibility pursuit-evasion game," *Journal of the Franklin Institute*, vol. 359, no. 18, pp. 10 420–10 454, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016003222007566

[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[11] A. Dobson, K. Solovey, R. Shome, D. Halperin, and K. E. Bekris, "Scalable asymptotically-optimal multi-robot motion planning," in *Proc. International Symposium on Multi-Robot and Multi-Agent Systems*, 2017.

[12] J. W. Durham, A. Franchi, and F. Bullo, "Distributed pursuit-evasion without mapping or global localization via local frontiers," *Autonomous Robots*, vol. 32, pp. 81–95, 2012.

[13] S. W. Feng, S. D. Han, K. Gao, and J. Yu, "Efficient algorithms for optimal perimeter guarding," in *Proc. Robotics: Science and Systems*, 2019.

[14] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view." *International Journal of Robotics Research*, vol. 25, no. 4, pp. 299–315, 2006.

[15] B. Gipson, M. Moll, and L. E. Kavraki, "Resolution independent density estimation for motion planning in high-dimensional spaces," in *Proc. IEEE International Conference on Robotics and Automation*, 2013, pp. 2437–2443.

[16] L. Gregorin, S. Givigi, E. Freire, E. Carvalho, and L. Molina, "Heuristics for the multi-robot worst-case pursuit-evasion problem," *IEEE Access*, vol. 5, pp. 17 552–17 566, Aug. 2017.

[17] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment," *International Journal on Computational Geometry and Applications*, vol. 9, no. 5, pp. 471–494, 1999.

[18] Y. C. Ho, A. Bryson, and S. Baron, "Differential games and optimal pursuit-evasion strategies," *IEEE Trans. Automatic Control*, vol. 10, pp. 385–389, 1965.

[19] R. Isaacs, *Differential Games*. New York: Wiley, 1965.

[20] V. Isler, N. Noori, P. Plonski, A. Renzaglia, P. Tokekar, and J. V. Hook, "Finding and tracking targets in the wild: Algorithms and field deployments," in *Proc. IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2015.

[21] L. Jaillet, J. Cortes, and T. Simeon, "Transition-based rrt for path planning in continuous cost spaces," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 2145–2150.

[22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[23] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robotics and Automation*, vol. 12, no. 4, pp. 566–580, Jun. 1996.

[24] A. Kolling and S. Carpin, "Multi-robot pursuit-evasion without maps," in *Proc. IEEE International Conference on Robotics and Automation*, 2010, pp. 3045–3051.

[25] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 2000.

[26] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 673–692, 2004.

[27] A. Q. Li, F. Amigoni, R. Fioratto, and V. Isler, "A search-based approach to solve pursuit-evasion games with limited visibility in polygonal environments," in *Proc. International Conference on Autonomous Agents and Multiagent Systems*, 2018, pp. 1693–1701.

[28] Y. Li, Z. Littlefield, and K. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," in *Proc. Workshop on the Algorithmic Foundations of Robotics*, 04 2015, pp. 263–282.

[29] E. Lozano, U. Ruiz, I. Becerra, and R. Murrieta-Cid, "Surveillance and collision-free tracking of an aggressive evader with an actuated sensor pursuer," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6854–6861, 2022.

[30] N. Mimmo, P. Bernard, and L. Marconi, "Avalanche victim search via robust observers," in *IEEE Trans. on Control Systems Technology*, vol. 29, no. 4, 2021, pp. 1450–1461.

[31] R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. Erkmen, *Search and Rescue Robotics*. Springer Berlin Heidelberg, 01 2008, pp. 1151–1173.

[32] T. Olsen, N. Stiffler, and J. M. O'Kane, "Robust-by-design plans for multi-robot pursuit-evasion," in *Proc. IEEE International Conference on Robotics and Automation*, 2022.

[33] T. Olsen, A. M. Tumlin, N. M. Stiffler, and J. M. O'Kane, "A visibility roadmap sampling approach for a multi-robot visibility-based pursuit-evasion problem," in *Proc. IEEE International Conference on Robotics and Automation*, 2021.

[34] T. D. Parsons, "Pursuit-evasion in a graph," in *Theory and Application of Graphs*, Y. Alavi and D. R. Lick, Eds. Berlin: Springer-Verlag, 1976, pp. 426–441.

[35] N. N. Petrov, "A problem of pursuit in the absence of information on the pursued," *Differentsial'nye Uraveniya (Differential Equations)*, vol. 18, pp. 1345–1352, 1982.

[36] ——, "The cossack-robber differential game," *Differentsial'nye Uraveniya (Differential Equations)*, vol. 19, pp. 1366–1374, 1983.

[37] A. D. Rahul Shome, Kiril Solovey, D. Halperin, and K. Bekris, "drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning," *Autonomous Robots*, vol. 44, pp. 443–467, 03 2020.

[38] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning," *International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016. [Online]. Available: https://doi.org/10.1177/0278364915615688

[39] N. M. Stiffler and J. M. O'Kane, "A complete algorithm for visibility-based pursuit-evasion with multiple pursuers," in *Proc. IEEE International Conference on Robotics and Automation*, 2014.

[40] ——, "A sampling based algorithm for multi-robot visibility-based pursuit-evasion," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[41] ——, "Complete and optimal visibility-based pursuit-evasion," *International Journal of Robotics Research*, vol. 36, Jul. 2017.

[42] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on Computing*, vol. 21, no. 5, pp. 863–888, Oct. 1992.

[43] P. Tokekar, D. Bhadauria, A. Studenski, and V. Isler, "A robotic system for monitoring carp in Minnesota lakes," *Journal of Field Robotics*, vol. 27, no. 3, pp. 681–685, 2010.

[44] B. Tovar and S. M. LaValle, "Visibility-based pursuit-evasion with bounded speed," *International Journal of Robotics Research*, vol. 27, pp. 1350–1360, 2008.

[45] M. Xanthidis, J. M. Esposito, I. Rekleitis, and J. M. O'Kane, "Motion planning by sampling in subspaces of progressively increasing dimension," *Journal of Intelligent and Robotic Systems*, 2020.

[46] A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, “Dynamic-domain rrts: Efficient exploration by controlling the sampling domain,” in *Proc. IEEE International Conference on Robotics and Automation*, 2005, pp. 3856–3861.