A Visibility-Based Pursuit-Evasion Problem

Leonidas J. Guibas Jean-Claude Latombe Steven M. LaValle David Lin Rajeev Motwani

Computer Science Department

Stanford University

Stanford, CA 94305

{guibas,latombe,lavalle,dlin,rajeev}@cs.stanford.edu

Abstract

This paper addresses the problem of planning the motion of one or more pursuers in a polygonal environment to eventually "see" an evader that is unpredictable, has unknown initial position, and is capable of moving arbitrarily fast. This problem was first introduced by Suzuki and Yamashita. Our study of this problem is motivated in part by robotics applications, such as surveillance with a mobile robot equipped with a camera that must find a moving target in a cluttered workspace.

A few bounds are introduced, and a complete algorithm is presented for computing a successful motion strategy for a single pursuer. For simply-connected free spaces, it is shown that the minimum number of pursuers required is $\Theta(\lg n)$. *For multiply-connected free spaces, the bound is* $\Theta(\sqrt{h}+\lg n)$ *pursuers for a polygon that has* n *edges and* h *holes. A set of problems that are solvable by a single pursuer and require a linear number of recontaminations is shown. The complete algorithm searches a finite graph that is constructed on the basis of critical information changes. It has been implemented and computed examples are shown.*

1 Introduction

The general problem addressed in this paper is an extension or combination of problems that have been considered in several contexts. Interesting results have been obtained for pursuit-evasion in a graph, in which the pursuers and evader can move from vertex to vertex until eventually a pursuer and evader lie in the same vertex [17, 20]. The *search number* of a graph refers to the minimum number of pursuers needed to solve a pursuit-evasion problem, and has been closely related to other graph properties such as cutwidth [16, 18]. It has also been shown that a graph can be searched monotonically (i.e., without recontamination) in [2, 12]. Pursuit-evasion scenarios in continuous spaces have arisen in a variety of applications such as air traffic control [1], military strategy [11], and trajectory tracking [10]. This has resulted in the formal study of general decision problems in which two decision makers have diametrically opposing interests. Classical pursuit-evasion games express differential motion models for two opponents, and conditions of capture or optimal strategies are sought [11]. For example, in the classical Homicidal Chauffeur game, conditions of inevitable collision can be expressed in terms of the nonholonomic turning-radius constraints of the pursuer and evader. Although interesting decision problems arise through the differential motion models, geometric free-space constraints are usually not considered in classical pursuit-evasion games. Once these constraints are introduced, the problem inherits the additional complications that arise in geometric motion planning.

A region of capture is often associated with a pursuit-evasion problem, and the "capture" for our problem is defined as having the evader lie within a line-of-sight view from a pursuer. A moving visibility polygon in a polygonal environment adds geometric information that must be utilized, and also leads to connections with the static art gallery problems [19, 22]. In the limiting case, art gallery results serve as a loose upper bound on the number of pursuers by allowing a covering of the free space by static guards, guaranteeing that any evader will be immediately visible. Far fewer guards are needed when they are allowed to move and search for an evader; however, the required motion strategies can become quite complex. A closely related art gallery variant is the watchman tour problem [5]. In this case a minimum-length closed path is computed such that any point in the polygon is visible from some point along the path. In our case, however, the pursuers have the additional burden of ensuring that an evader cannot "sneak" to a portion of the environment that has already been explored. The problem that we consider and other variations have been considered previously in [6, 23]. It was stated in [22] that it remained an interesting challenge to determine if a polygon is searchable by a single pursuer.

Several applications can be envisioned for problems and motion strategies of this type. For example, suppose a building security system involves a few mobile robots with cameras or range sensors that can detect an intruder. A patrolling route can be automatically computed that guarantees that any mobile intruder will eventually be found. To optimize expenses, it would also be important to know the minimum number of robots that would be needed. Applications are not necessarily limited to adversarial targets. For example, the task might be to automatically locate another mobile robot, items in a warehouse or factory that might get moved during the search process, or possibly even people in a search/rescue effort. Such strategies could be used by automated systems or by human searchers.

Section 2 presents a precise mathematical formulation of the problem. Section 3 presents several bounds on the number of required pursuers and related problems. Section 4 presents general concepts for reducing the problem to a finite graph search and a complete algorithm for computing a solution strategy for a given free space. Section 5 shows several example solution strategies that were computed using our implemented algorithm, and discusses some of the practical implementation issues. Conclusions are presented in Section 6.

2 Problem Definition

The pursuers and evader are modeled as points that move in a polygonal free space, F. Let $e(t) \in F$ denote the position of the *evader* at time $t \geq 0$. It is assumed that $e : [0, \infty) \to F$ is a continuous function, and the evader is capable of moving arbitrarily fast. The initial position $e(0)$ and the path e are assumed unknown to pursuers. Any region in F that might contain the evader will be referred to as *contaminated*; otherwise it will be referred to as *cleared*. If a region is contaminated, becomes cleared, and then becomes contaminated again, it will be referred to as *recontaminated*.

Let $\gamma^i(t)$ denote the position of the i^{th} *pursuer* at time $t \geq 0$. Let γ^i represent a continuous path of the i^{th} pursuer of the form $\gamma^i : [0, \infty) \to F$. Let γ denote a *(motion) strategy*, which refers to the specification of a continuous path for every pursuer: $\gamma = \{\gamma^1, \ldots, \gamma^N\}$.

For any point, $q \in F$, let $V(q)$ denote the set of all points in F that are visible from q (i.e., the linear segment joining q and any point in $V(q)$ lies in F). A strategy, γ, is a *solution strategy* if for every continuous function $e : [0, \infty) \to F$ there exists a time $t \in [0, \infty)$ and an $i \in \{1, \ldots, N\}$ such that $e(t) \in V(\gamma^i(t))$. This implies that the evader will eventually be seen by one or more pursuers, regardless of its path. Let $H(F)$ represent the minimum number of pursuers for which there exists a solution strategy for F.

Section 3 presents some bounds on $H(F)$ for classes of free spaces, and also shows that some polygons for which $H(F) = 1$ only admit solutions in which a region becomes contaminated a linear number of times, regardless of the initial placement of the pursuers. Section 4 addresses the problem of computing a solution strategy, γ, for a given F.

3 Worst-Case Bounds

Several new bounds are presented in this section. For a simply-connected free space, F, with n edges, it is shown that $H(F) = \Theta(\lg n)$. For a free space, F, with h holes, it is shown that $H(F) = \Theta(\sqrt{h} + \lg n)$. For the case of problems in which $H(F) = 1$, it is shown that the same region can require recontamination as many as $\Omega(n)$ times. This result is surprising because pursuit-evasion in a graph is known not to require any recontaminations [12]. In [23] a free space was given that requires two recontaminations, which at least established that recontamination is generally necessary for visibility-based pursuit evasion.

Consider the problem of determining the minimum number of pursuers, $H(F)$, required to find an evader in a given free space F. This number will generally depend on both the topological and geometric complexity of F. In [23] a class of simple polygons is identified for which a single pursuer suffices (referred to as "hedgehogs"). Some interesting upper and lower bounds on $H(F)$ are presented in terms of free space properties such as "bushiness" and reflex vertices in [25]. For any F that has at least one hole, it is clear that at least two pursuers will be necessary; if a single pursuer is used, the evader could always move so that the hole is between the evader and pursuer. In some cases subtle changes in the geometry significantly affect $H(F)$. Consider for example, the problems in Figure 1. Although the problems are similar, only the problem in the lower right requires two pursuers.

Consider $H(F)$ for the case of simply-connected free spaces. Let n represent the number of edges in the free space, which is represented by a simple polygon in this case. A logarithmic worst-case bound can be established:

Theorem 1 *For any simply-connected free space F with n edges, $H(F) = O(\lg n)$.*

Proof: The proof is built on the following observation. Suppose that two vertices of F are connected by a linear segment, thus partitioning F into two simply-connected, polygonal

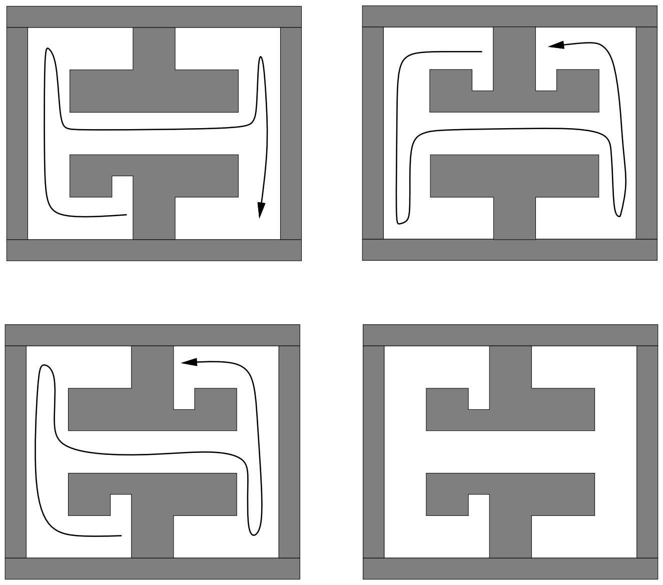
Figure 1: Four examples are shown that have similar geometry. The example in the lower right requires two pursuers, while the other three examples require only one.

components, F_1 and F_2. If $H(F_1) \leq k$ and $H(F_2) \leq k$ for some k, then $H(F) \leq k+1$ because the same k pursuers can be used to clear both F_1 and F_2. This requires placing a static $(k+1)^{th}$ pursuer at the edge common to F_1 and F_2 to keep F_1 cleared after the k pursuers move to F_2 (assuming arbitrarily that F_1 is cleared first).

In general, if two simply-connected polygonal regions share a common edge and can each be cleared by at most k pursuers, then the combined region can be cleared at most $k+1$ pursuers. Recall that for any simple polygon, a pair of vertices can always be connected so that polygon is partitioned into two regions, each with at least one third of the edges of the original polygon [4]. This implies that F can be recursively partitioned until a triangulation is constructed, and each triangular region only requires $O(\lg n)$ recombinations before F is obtained (i.e., the recursion depth is logarithmic in n). Based on the previous observation and the fact that each triangular region can be trivially searched by a single pursuer, $H(F) = O(\lg n)$. $\Box$

A similar logarithmic bound was also obtained in [25]. The remaining question for simply-connected free spaces is whether there actually exist problems that require a logarithmic number of pursuers. Some results from graph searching will first be described and utilized to construct difficult worst-case problem instances. Let *Parsons' problem* refer to the graph-

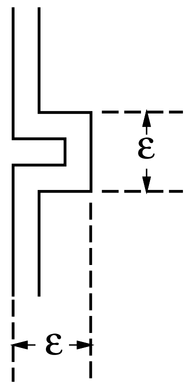

Figure 2: A corridor of this shape disconnects second-order visibility between the two entrances, and can be used to construct geometric equivalents of Parsons' problem for planar graphs.

searching problem presented in [17, 20]. The task is to specify the number of pursuers required to find an evader that can execute continuous motions along the edges of a graph. Instead of using visibility, capture is achieved when one of the pursuers "touches" the evader. Let G represent a graph, and $S(G)$ represent the number of needed pursuers, referred to as the search number of G.

The following lemma implies that a geometric realization of any planar graph instance can be constructed:

Lemma 1 *For every planar graph, G, there exists a polygonal free space F such that Parsons' problem on G is equivalent to the visibility-based pursuit evasion problem on F.*

Proof: Since G is planar, a geometric representation exists in the plane in which points in $\Re^2$ correspond to vertices in G, and linear segments between the points correspond to edges in G. Consider the corridor structure shown in Figure 2. Every linear segment in the geometric representation of G can be replaced by a corridor of sufficient length as shown in Figure 2. Furthermore, there exists an $\epsilon > 0$ such that no pair of corridors intersect, except near the points that correspond to vertices of G. Portions of the corridor edges can be removed at corridor junctions to prevent overlap. Let F refer to the resulting polygon, which represents a network of bent corridors.

The next task is to show that searching F is equivalent to searching G. Recall that any successful Parsons' search strategy can be specified by the traversal of a sequence of edges for each pursuer. If the sequence of bent corridors is explored that corresponds to the edges of a solution strategy for G, then F will be successfully searched. This is true since using visibility to "see" an evader in a corridor is at least as powerful as attempting to "touch" an evader in a continuous graph edge.

Next consider whether any solution strategy for F can be used to equivalently search G. The corridor piece from Figure 2 is intentionally bent in four places, which causes all advantages of visibility to be lost. Suppose this corridor is connected at both ends to other corridors, and that pursuers are placed at each end (at positions q^1 and q^2). For this corridor, $V(V(q^1))$ and $V(V(q^2))$ are disjoint; $V(V(q))$ represents the set of all points from which at least one point in $V(q)$ is visible. Although both pursuers can see into the corridor, one of the pursuers must travel into the middle of the corridor at some point to search for the evader. It must travel far enough so that the entrance to the corridor is no longer visible (leaving the entrance "unguarded"). The central portion of each corridor will correspond directly to an edge in G, since a central portion (edge) can be explored only by leaving a junction (vertex) unguarded.

Consider any given motion strategy γ that is a solution for F. Suppose there are N pursuers. For each $i \in \{1, \ldots, N\}$, it will be shown that γ_i can be used to determine a sequence of edges for the i^{th} pursuer in G. Without loss of generality, it can be assumed that γ_i only traverses the centers of the corridors (i.e. equal distance is maintained between the corridor walls). Thus, γ_i can be characterized by indicating how far into a corridor the i^{th} pursuer travels, at which point it performs a reversal, which corridor it selects at a junction, etc. Suppose γ_i travels from junction to junction, with reversals only being made at junctions. In this case, every corridor in F that is cleared will cause the corresponding edge in G to be cleared. If γ is a solution strategy for this case, then a corresponding solution strategy for G is implied. Suppose that γ_i actually causes reversals to occur in a corridor (i.e., not at a junction). If the pursuer changes direction but still traverses the full length of the corridor (it must change direction at least twice), then the corresponding edge in G will still be cleared. If the pursuer returns to the originating junction, then there are two possible cases. If the pursuer travels far enough to clear the central portion of the corridor, then the originating junction is left unguarded. The corresponding edge in G can be cleared by moving the pursuer from the originating vertex (which corresponds to the originating junction in F), across the edge in G to the edge's other vertex, and back to the originating vertex. If the pursuer does not travel far enough into the corridor to clear the central portion, then this portion of γ^i does not make progress, and can be discarded (i.e., no corresponding strategy portion needs to be considered for G). Thus any solution strategy, γ, for F can be used to determine a corresponding solution strategy for G. $\square$

A theorem from [20] will be useful for proving Theorem 2, which provides a logarithmic lower bound on the number of pursuers needed to successfully search a simply-connected free space:

Lemma 2 (Parsons) *Let G be a tree. Then $S(G) \geq N+1$ if and only if there exists a vertex in G whose removal separates G into at least three components, G_1, G_2, and G_3, such that $S(G_i) \geq N$ for $i \in \{1, 2, 3\}$.*

Theorem 2 *There exist simply-connected free spaces F with n edges such that $H(F) = \Omega(\lg n)$.*

Proof: Using Lemma 2, a tree, G, can be constructed recursively that has a constant branching factor of three, height $N-1$, and requires N pursuers (an example is given in [20]). By Lemma 1, an equivalent geometric instance can be constructed for each N. Figure 3 depicts these geometric instances, for which $H(F) = \Omega(\lg n)$ $\square$

Theorem 1 and Theorem 2 together imply a tight logarithmic bound, $H(F) = \Theta(\lg n)$.

Next consider the class of problems for which F has h holes. Both upper and lower bounds are established on $H(F)$ which are proportional to $\sqrt{h}$.

Theorem 3 *For any free space F with n edges and h holes, $H(F) = O(\sqrt{h} + \lg n)$.*

Proof: Divide the pursuers into two groups: $O(\sqrt{h})$ pursuers will be used to reduce the polygon to simply-connected components, and $O(\lg n)$ pursuers will be used to clear each component. Construct an arbitrary triangulation of F. Let a *trichromatic triangle* be defined as a triangle that touches three distinct connected components of the boundary of F. Using

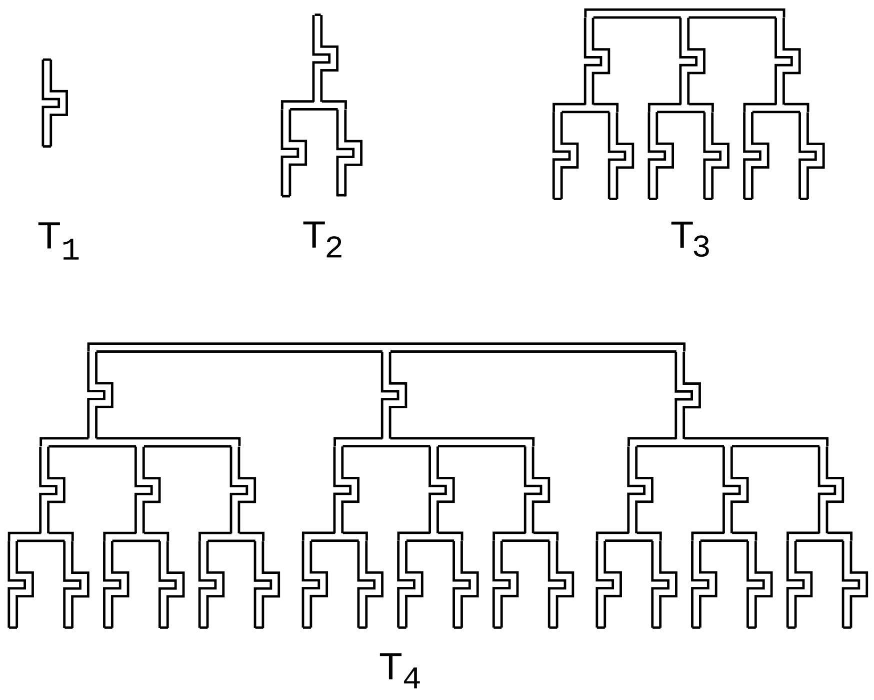

Figure 3: Systematic construction of simply-connected free spaces that require $\Omega(\lg n)$ pursuers.

at most $O(h)$ trichromatic triangles, F can be partitioned into simply-connected components whose boundaries are comprised of the boundary of F and edges of trichromatic triangles. To establish this, form a planar graph by placing a vertex in each hole of F and one vertex outside of F. For every trichromatic triangle edge joining two boundaries, form an edge for this graph by joining the vertices corresponding to these boundaries by a path along the trichromatic edge, in the obvious way. From the planarity of this graph we can easily argue that the overall number of trichromatic edges, and therefore of trichromatic triangles, is $O(h)$.

Consider the dual graph of the triangulation, which has $O(n)$ edges and vertices. Take the subgraph induced by taking only the vertices that correspond to trichromatic triangles in the original triangulation. The planar graph separator theorem [15] implies that at most $O(\sqrt{h})$ edges can be chosen to partition the graph into two portions with at least one third of the edges on each side of the partition. Each edge in the induced subgraph corresponds to a simply-connected region of F that can be cleared with $O(\lg n)$ pursuers by Theorem 1. In fact, the same set of pursuers can be used for each simply connected component. The $O(\sqrt{h})$

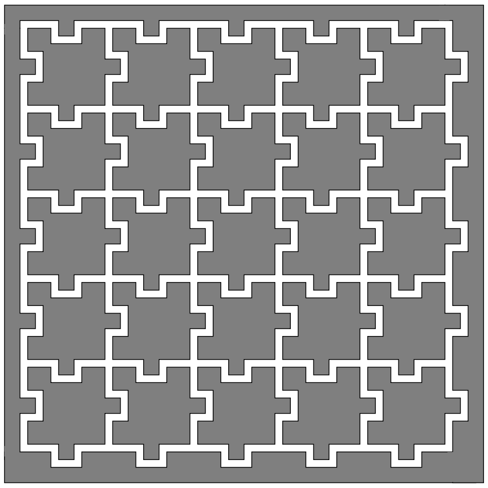

Figure 4: An instance from a sequence of problems that requires a number of pursuers that is at least proportional to the square root of the number of holes.

pursuers then form a barrier that maintains the cleared areas obtained by other pursuers on either side, much in the same way as the partitioning edges in the proof of Theorem 1. The planar graph separator theorem can be applied recursively to the remaining portions of F on either side of a barrier, and the free space can be cleared using the same progression as for Theorem 1. At the i^{th} level of recursion, at most $\frac{2}{3}$ as many pursuers will be needed to form a barrier in comparison to the $(i-1)^{th}$ level of recursion. The free space is reduced to simply-connected components that can be cleared using $O(\lg n)$ pursuers. The total number of pursuers needed to form barriers is $O(\sqrt{h} + \sqrt{\frac{2}{3}h} + \sqrt{\frac{4}{9}h} + \cdots) = O(\sqrt{h})$. Thus, F can be cleared using at most $O(\sqrt{h} + \lg n)$ pursuers. $\square$

Theorem 4 *There exist free spaces F with n edges and h holes such that $H(F) = \Omega(\sqrt{h} + \lg n)$.*

Proof: For any positive integer k, a planar graph of cutwidth k can be constructed using $O(k^2)$ vertices and edges. Recall that the cutwidth, $CW(G)$, is the minimum cutwidth taken over all possible linear layouts of G. A linear layout of G is a one-to-one function mapping the vertices of G to integers, and the cutwidth for a particular layout is the maximum over all i of the number of edges connecting vertices assigned to integers less than i to vertices assigned to integers as large as i. Define a sequence of planar graphs, $G_1, G_2, \ldots$. Let the vertices of G_k correspond to the set of all points with integer coordinates, (i, j), such that

$0 \leq i, j \leq k$. Let G_k connect vertices v and w by an edge if and only if v adn w are distance 1 apart (i.e., a standard four-neighborhood). The cutwidth of G_k is k.

It is established in [16] that for all graphs G, the search number $S(G)$ is related by $S(G) \leq CW(G) \leq \lfloor deg(G)/2 \rfloor \cdot S(G)$, in which $deg(G)$ is the maximum vertex degree of G. Because $deg(G_k) = 4$, $S(G) \leq k \leq 2S(G)$. Using Lemma 1, geometric instances of G_k such as the one shown in Figure 4 can be constructed. Both G_k and each geometric instance require $\Omega(k)$ pursuers. There is a quadratic number of holes in each geometric instance; hence, $H(F) = \Omega(\sqrt{h})$. This corridor structure can be combined with the structure from Theorem 2 to yield an example that requires $\Omega(\sqrt{h} + \lg n)$ pursuers. $\square$

Theorem 4 and Theorem 3 together imply a tight bound, $H(F) = \Theta(\sqrt{h} + \lg n)$.

The final theorem of this section pertains to the case of free spaces that can be searched by a single pursuer. A similar result is also obtained in [6]. It states that there exist examples that require recontaminating some portion of the free space a linear number of times. This result is surprising because for Parsons' problem it was shown in [12] that no recontamination is necessary (a shorter proof of this appears in [2]). Theorem 5 establishes that a linear number of recontaminations can be needed, and it still remains open to determine whether the number of recontaminations can be bounded from above by a polynomial, which would imply that the problem of deciding whether $H(F) = 1$ lies in NP.

Theorem 5 *There exists a sequence of simply-connected free spaces with* $H(F) = 1$ *such that* $\Omega(n)$ *recontaminations are required for* n *edges.*

Proof: It will be shown that the example in Figure 5 requires $k - 2$ recontaminations by visiting the point $a \in F$ a total of $k - 1$ times to repeatedly clear the "peak." Without loss of generality, consider the set of strategies that can be specified by identifying the sequence of points, $a, b_1, \ldots, b_k, c_1, \ldots, c_k$, that are visited. Assume that the shortest-distance path is taken between any pair of points.

We claim that if all legs are initially contaminated, then they must be visited in one of two orders: $(b_1, c_1, b_2, c_2, \ldots, b_k, c_k)$ or $(c_k, b_k, c_{k-1}, b_{k-1}, \ldots, c_1, b_1)$. To refute this claim, consider visiting b_i for some $1 < i < k$, followed by a visit to another "leg", say b_j (or c_j).

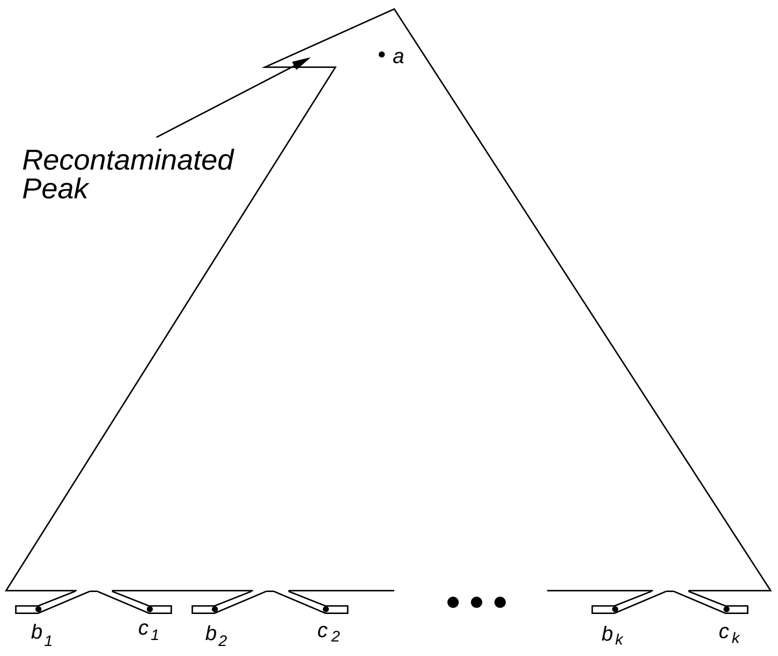

Figure 5: A linear number of recontaminations is required. Although this polygon can be searched by a single pursuer, the peak must be visited $k-1$ times.

If any legs between b_i and b_j are contaminated, then b_i will become recontaminated. In general, any legs that are cleared will be recontaminated if a non-neighboring leg is cleared next. This prevents the pursuer from making progress toward clearing the entire free space. Thus, the legs must be explored in order.

Because of symmetry, consider visiting the legs from left to right without loss of generality. Assume that the peak is initially contaminated. The points b_1 and c_1 can be visited to clear the leftmost set of legs; however, these will get contaminated when b_2 is visited. By traveling from c_1 to a to b_2, the leftmost set of legs remain cleared because the peak is cleared. When c_2 is visited, the leftmost three legs remain cleared; however, the peak becomes recontaminated. Thus, a will have to be visited again before clearing b_3. By induction on i for $1 < i \leq k$, the peak will have to be cleared by visiting a each time between visits to c_i and b_{i+1}. This implies that a will be visited $k-1$ times, resulting in $k-2$ recontaminations. $\square$

4 Computing a Solution Strategy

While Section 3 addressed worst-case bounds over certain problem classes, this section covers concepts and algorithms for computing a solution strategy for a given free space. NP-hardness is established in Section 4.1. Section 4.2 defines an information space for a single pursuer, and provides a general method for handling this space combinatorially. Section 4.3 presents a complete algorithm for the case in which $H(F) = 1$ that computes a solution strategy by decomposing F into convex cells that each prevent critical changes in information. This algorithm is quite efficient in practice, and was used to compute the examples shown in Section 5. The case in which $H(F) > 1$ is discussed in Section 4.4.

4.1 Complexity of the General Problem

A *complete* algorithm must compute a solution strategy for a given number of pursuers, if such a strategy exists. It is natural to compare the notion of completeness for this problem to completeness for the basic motion planning problem (i.e., the algorithm will find a collision-free path if such a path exists [3]). One important difference, however, is that the *minimum* number of pursuers is crucial, but does not have a correspondence for the basic path planning problem. A variety of simple, heuristic algorithms can be developed that require more pursuers than necessary (for example, place a static pursuer inside of each cell of a convex decomposition of F). The problem becomes most difficult when the minimum number of pursuers is requested.

The problem of determining the minimum number of pursuers is intractable if $P \neq NP$:

Theorem 6 *Computing $H(F)$ is NP-hard.*

Proof: It is shown in [18] that Parsons' problem for a planar graph with maximum vertex degree 3 is NP-complete (i.e., computing the search number, $S(G)$). By Lemma 1, equivalent geometric instances can be constructed, which implies that computing $H(F)$ is NP-hard. □

4.2 Identifying Critical Information Changes

During the execution of a strategy, the pursuers must identify the contaminated region. At a given time, this important piece of information generally depends on the initial positions of the pursuers and their history of past positions, up to the given time. As pursuers move, this information changes continuously; however, to develop a complete algorithm we will only be interested in tracking times in which the pursuers' information changes *combinatorially.* This section defines an information space that corresponds to the knowledge of the pursuers, and presents a general scheme for partitioning the information space into equivalence classes of information. This is inspired in part by a standard approach used in motion planning, which is to preserve completeness by using a decomposition of the configuration space that is constructed by analyzing critical events. For example, in [21] a cell decomposition is determined by analyzing the contact manifolds in a composite configuration space that is generated by the positions of several disks in the plane. Section 4.3 presents a complete algorithm that uses this concept for the case of $H(F) = 1$.

Assume that a search is performed by a single pursuer; the concepts in this section can be extended to multiple pursuers with minor adaptation [14]. Let $q \in F$ represent the current pursuer position. Let $S \subseteq F$ represent the set of all contaminated points in F. Let $\eta = (q, S)$ represent an *information state.* The set of all possible information states will be referred to as the *information space.* The information space is a standard representational tool for problems that have imperfect state information, and has been useful for other motion planning problems [7, 13].

Suppose that a strategy is parameterized with a time interval $t \in [0, t_f]$ for some fixed $t_f > 0$. For a fixed strategy, γ, and an initial set of contaminated points, $S(0)$, a path in the information space is obtained. At a given $0 < t \leq t_f$ the set of contaminated points, $S(t)$, can be determined from the history $\{\gamma(t') | t' \in [0, t]\}$. Let $\Psi(\eta, \gamma, t_0, t_1)$ represent the information state that will be obtained by starting from information state η, and applying the strategy γ from t_0 to t_1. The function Ψ can be thought of as a "black box" that produces the resulting information state when a portion of a given strategy is executed.

The next definition describes an information invariance property, which when satisfied allows the information space to be partitioned into equivalence classes. A connected set $D \subseteq F$ is *conservative* if $\forall \eta$ such that $q \in F$, and $\forall \gamma : [t_0, t_1] \rightarrow D$ such that γ is continuous and $\gamma(t_0) = \gamma(t_1) = q$, then the same information state, $\eta = \Psi(\eta, \gamma, t_0, t_1)$, is obtained. This implies that the information state cannot be altered by moving along closed paths in D. Just as in the case of motions in a conservative field, the following holds:

Lemma 3 *If D is conservative then for any two continuous paths, γ_1, γ_2, mapping into D such that $\gamma_1(t_0) = \gamma_2(t_0)$ and $\gamma_1(t_1) = \gamma_2(t_1)$ then $\Psi(\eta, \gamma_1, t_0, t_1) = \Psi(\eta, \gamma_2, t_0, t_1)$, for any η.*

Proof: Select any third continuous trajectory, $\gamma_3 : [t_0, t_1] \rightarrow D$, such that $\gamma_3(t_0) = \gamma_1(t_1)$ and $\gamma_3(t_1) = \gamma_1(t_0)$ (i.e., heading in the opposite direction). Form a new trajectory, γ_{132}, by concatenating the trajectories γ_1, γ_3, and γ_2 in the following manner:

$$\gamma_{132}(t) = \begin{cases} \gamma_1(3(t - t_1)) & \text{If } t < t_1 + \frac{t_2 - t_1}{3} \\ \gamma_3(3(t - t_1 - \frac{t_2 - t_1}{3})) & \text{If } t_1 + \frac{t_2 - t_1}{3} \leq t \leq t_1 + 2\frac{t_2 - t_1}{3} \\ \gamma_2(3(t - t_1 - 2\frac{t_2 - t_1}{3})) & \text{If } t > t_1 + 2\frac{t_2 - t_1}{3} \end{cases} .$$

Note that the information state does not depend on the speed of the pursuer. The resulting information state will be $\Psi(\eta, \gamma_1, t_0, t_1)$ because γ_3 followed by γ_2 forms a closed-loop path, and thus yields the same information state by conservativity of D. Note that γ_1 followed by γ_3 is also a closed-loop path, which implies that γ_2 must bring the information state from η to $\Psi(\eta, \gamma_1, t_0, t_1)$. Hence, $\Psi(\eta, \gamma_1, t_0, t_1) = \Psi(\eta, \gamma_2, t_0, t_1)$. $\square$

Thus, the information state that results from moving between $q_1 \in D$ and $q_2 \in D$ is invariant with respect to the chosen path, assuming the pursuer remains in D.

4.3 A Complete Algorithm for a Single Pursuer

Since the general problem is NP-hard, it is worth focusing on the complete algorithm for the case of a single pursuer. This implies that F is simply-connected, otherwise at least two pursuers would be required. The basic idea is to partition the free space into conservative regions, and perform a search on the resulting equivalence classes in the information space. This algorithm has been implemented and tested on a variety of examples; some of these examples are shown in Section 5.

Representing the information state A useful representation of the information state will be defined for the case of a single pursuer. Suppose the pursuer is at a position $q \in F$. Note that the edges in the visibility polygon, $V(q)$, generally alternate between being part of

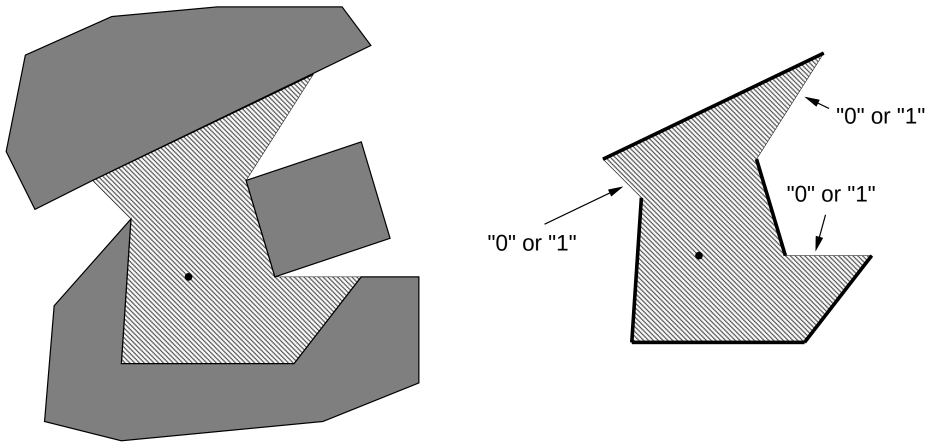

Figure 6: Binary labels can be used to encode the information state.

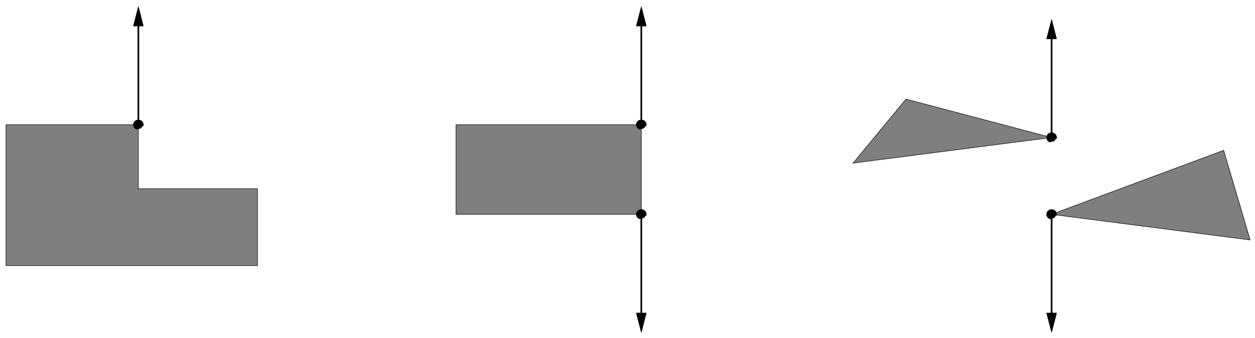

Figure 7: Ray shooting is performed for three general cases to form the conservative regions.

the boundary of F and crossing the interior of F. Let each edge that enters the interior of F be referred to as a *gap edge.* Consider associating a binary label with each gap edge. If the portion of the free space that borders the gap edge is contaminated, then it is assigned a "1" label; otherwise, it is assigned a "0" label, indicating that it is clear. Let $B(q)$ denote a binary vector of gap edge labels, with one label for each gap edge in the visibility polygon. Note that each component of the contaminated region is bounded by a simple polygon that must coincide with exactly one gap edge from $V(q)$ that has the "1" label. Thus, the specification of the pair $(q, B(q))$ uniquely characterizes the information state.

A decomposition of F into conservative regions Let $\mathcal{D}$ represent a collection of convex regions that define a partition of F. The boundaries of regions in $\mathcal{D}$ are obtained by extending rays from edges in F, and from certain pairs of vertices, as shown in Figure

7. Each edge is extended in any direction possible, and each pair of vertices is extended outward only if both directions are free along the line drawn through the pair. A similar decomposition has been used for robot localization in [9, 24], and generates $O(n^3)$ regions in the worst case.

The following statement yields an important property that will allow us to explore the information space combinatorially:

Lemma 4 *Each region in $D \in \mathcal{D}$ is conservative.*

Proof: Consider representing the information state using $(q, B(q))$, and let a pursuer move using any continuous, closed-loop path $\gamma : [t_0, t_1] \rightarrow D$. We intend to establish that $B(q)$ at time t_1 will be the same as at time t_0, regardless of the choice of γ. Recall that each label in $B(q)$ corresponds to a connected component of $F \setminus V(q)$.

Let $q_1, q_2 \in D$ denote two distinct pursuer positions. There is a one-to-one correspondence between gap edges in $V(q_1)$ and $V(q_2)$, and each corresponding edge pair, say E_1 and E_2, shares a common vertex. If this were not true, then a ray of the form indicated in Figure 7 would lie between q_1 and q_2, implying that q_1 and q_2 do not both lie in D.

Each gap edge borders a connected component of $F \setminus V(q)$, which may or may not be comtaminated. While following any continuous path $\gamma : [t_0, t_1] \rightarrow D$, the connected components of $F \setminus V(q)$ will deform; however, it is impossible to change to the topology of $F \setminus V(q)$ (i.e., the same number of connected components will be maintained). This implies that no components are split or merged; therefore, the gap edge for each component must retain the same label. Thus, it is impossible to change $B(q)$ by moving the pursuer inside of D. $\square$

Figure 8 illustrates how the regions are conservative by showing a single pursuer that is approaching the end of a corridor. If the closed-loop motion on the left is executed, the end of the corridor remains contaminated. This implies that although the information state changes during the motion, the original information state is obtained upon returning. During the closed-loop motion on the right, the gap edge disappears and reappears. In this case, the resulting information state is different. The gap edge label is changed from “1” to “0”.

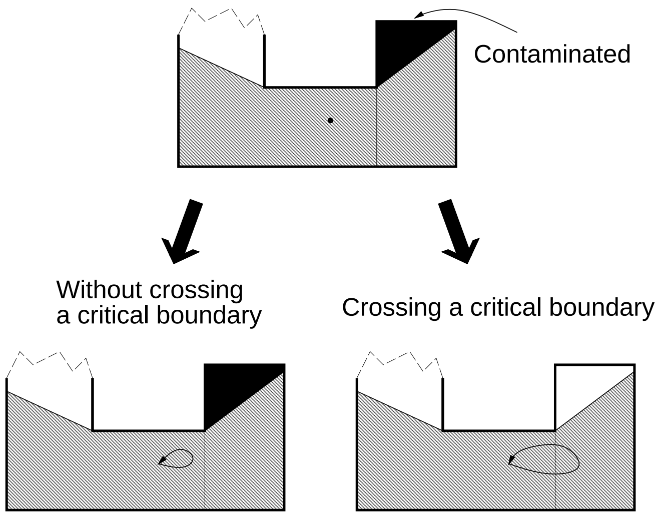
Contaminated
Without crossing a critical boundary
Crossing a critical boundary

Figure 8: A critical event in the information space can only occur when edge visibility changes.

A directed graph of information equivalence classes The regions in $\mathcal{D}$ and their natural adjacency relationships define a finite, planar graph, G. Each vertex of G corresponds to a region $D \in \mathcal{D}$ and an edge is defined if the boundaries of two regions have a one-dimensional intersection. A path in F can be constructed from a path in G by connecting centroids of adjacent regions in $\mathcal{D}$ by linear segments. Vertices in G must sometimes be visited multiple times in a solution strategy because $B(q)$ can be distinct each time. Initially, the pursuer will be in some position with all gap edges labeled with "1". The goal is to find any sequence of vertices in G that leaves the pursuer at some position with all gap edges labeled with "0".

A directed *information graph*, G_I, can be derived from G. For each vertex in G, a set of vertices are included in G_I, one for each possible labeling of the gap edges. For example, suppose a vertex in G represents some region D, and there are 2 gap edges for $B(q)$ at any $q \in D$. Four vertices will be included in G_I that all correspond to the pursuer in D; however, each vertex represents a unique possibility for $B(q)$: "00", "01", "10", or "11". Let a vertex in G_I be identified by specifying the pair $(q, B(q))$.

To complete the construction of G_I, the set of edges must be defined. This requires determining the appropriate gap edge labels as the pursuer changes regions. This in turn requires certain correspondences between gap edges to be made in advance, as indicated in

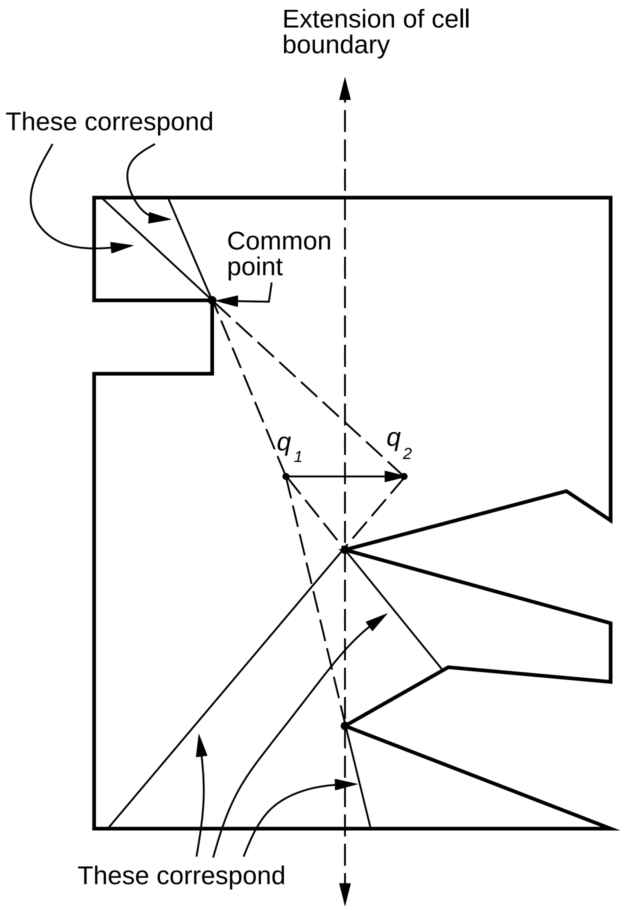

Figure 9: The correspondences between gap edges from different neighboring conservative regions can be directly determined. The information states are updated when moving between regions by using this correspondence.

Figure 9. Suppose the pursuer moves from $q_i \in D_i$ to $q_j \in D_j$. For the simple case shown in the lower right of Figure 8, assume that the gap edge on the left initially has a label of "0" and the gap edge on the right has a label of "1". Let the first bit denote the leftmost gap edge label. The first transition is from "01" to "0", and the second transition is from "0" to "00". The directed edges in G_I are given by the transition from $(q_i,\text{“01”})$ to $(q_j,\text{“0”})$ and from $(q_j,\text{“0”})$ to $(q_i,\text{“00”})$.

There are four possible cases that occur in general: a gap edge disappears, a gap edge appears, two or more gap edges merge into one, a gap edge splits into two or more. If a gap edge appears, it always receives a "0" label. If any n gap edges are merged, the merged gap edge will receive a "1" label if any of the original gap edges have a "1" label. If a gap edge splits, the new gap edges will obtain the value of the original gap edge.

This completes the definition of a directed graph that captures the information space combinatorially. Any vertex in G_I of the form $(q, B(q))$ such that $B(q) =$ "$00\cdots0$" represents a goal vertex. The initial vertex can be given, or selected arbitrarily such that $B(q) =$ "$11\cdots1$". A standard graph search algorithm can be employed to find a path in G_I between initial and goal vertices. The next theorem establishes the completeness of this approach.

Theorem 7 *An algorithm that will find any path to a goal vertex from an initial vertex in G_I is complete for the visibility-based pursuit-evasion problem in the case of $H(F) = 1$.*

Proof: The algorithm is complete if the existence of any solution strategy implies that a path exists in G_I between initial and goal vertices. Let $\{D_1, \ldots, D_n\}$ denote the sequence of regions in $\mathcal{D}$ that are traversed by any given solution strategy, γ. Each D_i corresponds to a vertex in G, and $\{D_1, \ldots, D_n\}$ corresponds to a path in G. This in turn corresponds to a path in G_I. By Lemma 3 and Lemma 4, the information state does not depend on the path chosen within each region, D_i. From this and the fact that γ is a solution strategy, the vertex obtained at the end of the corresponding path in G_I is a goal vertex. $\Box$

4.4 Multiple pursuers

In general, the conservative region concept that was presented in Section 4.1 can be applied to yield a decomposition of the $2N$-dimensional space that encodes the positions of the pursuers. Due to the hardness of the general problem, however, it still remains challenging to develop and implement a practical algorithm even for the case of two pursuers. A decomposition must be constructed for the case of N pursuers such that the edges in the union of the N visibility polygons do not change (edges to not vanish or appear). Conservative regions were formed for the case of $H(F) = 1$ by avoiding the critical changes in visibility (Figure 7), and one would hope that a Cartesian product of planar regions could be formed to directly define conservative regions for multiple pursuers; however, as Figure 10 indicates, the conservative regions can be considerably more complicated for the two pursuer case. Gap edges from two different visibility polygons can intersect in such a way that it is possible to execute a closed-loop path that changes the information state while keeping both pursuers within their

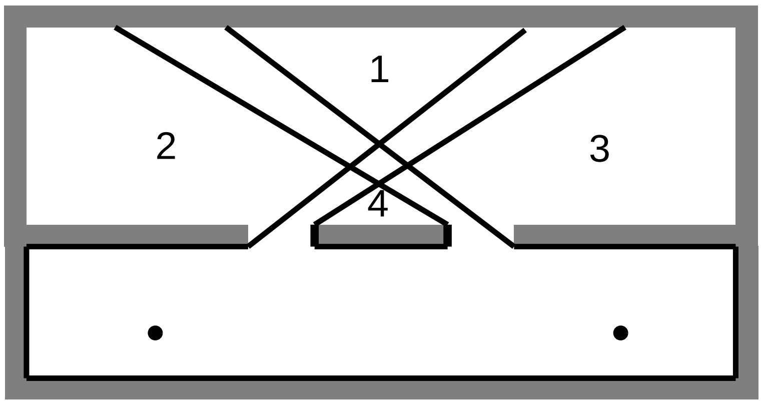

Figure 10: This example shows that the information state can be changed by a closed loop path without crossing a critical boundary, of the type shown in Figure 7. There are two gap edges for each pursuer; however, each gap edge can maintain two different labels. For example, Region 2 could be clear, while Region 4 could be contaminated. Furthermore, Region 1 can be cleared without causing any gaps edges to change critically (for example, by moving the left robot toward the right and returning).

regions in $\mathcal{D}$. For the example in Figure 10, suppose the pursuer on the left moves toward the right and returns, while the pursuer on the right remains stationary. This will cause Region 1 to be cleared. This constraint defines a three-dimensional algebraic manifold in $\Re^4$. The algebraic constraints that correspond to these types of cases significantly increase the implementation difficulty and add numerous conservative regions, which will dramatically hinder efficiency.

5 Computed Examples

The algorithm from Section 4.3 is implemented in C++ and executed on an SGI Indigo2 workstation with a 200 Mhz MIPS R4400 processor. The computation times and other parameters for several examples are listed in Figure 11. The implementation uses the quad-edge structure from [8] to maintain the topological ordering of the regions. The information graph G_I is searched using Dijkstra's shortest path algorithm, where the edge cost is taken as the distance between adjacent cell centroids. The solution is computed by traversing from cell centroid to cell centroid, causing the computed path for the pursuer to be jagged in most

Problem	Edges	Nodes in G_c (Cells)	Nodes in G_I (Information)	Precomp. Time (sec)	Searching Time (sec)	Total Time
Fig. 12	28	25	200	0.04	0.02	0.06
Fig. 13	68	130	1727	0.44	0.12	0.56
Fig. 14	46	237	8787	0.53	1.59	2.12
Fig. 15	65	246	18830	0.87	9.86	10.73
Fig. 16	70	888	103049	3.00	168.63	171.63

Figure 11: Various statistics are shown for the computed examples.

cases. In some applications, it might be appropriate to employ smoothing algorithms to the path to respect additional problem constraints.

Figures 12-16 show several computed examples. Due to a large number of conservative cells, Figures 14-16 are illustrated with the cell decompositions in separate diagrams from the solution diagrams. Figure 14 shows the hookpin example described in [23]. Note that the leftmost pin is recontaminated twice, and the pins are visited in the same order as mentioned in [23]. Figure 15 is an instance of the sequence described in Section 3 that requires a linear number of recontaminations. The region near the top of the figure is recontaminated 3 times. The final example generated a large number of conservative cells, which significantly increased computation time.

6 Conclusions

A visibility-based planning problem has been identified in this paper that involves searching for an unpredictable evader in a polygonal environment. This task can represent a basic operation in a variety of robotic applications, such as surveillance with mobile robots. Other potential applications include search-rescue operations and military strategy.

Several bounds were obtained. A tight logarithmic bound on the number of needed pursuers was shown for the case of a simply-connected free space. A tight square-root bound was expressed in terms of the number of holes for multiply-connected free spaces. A few open problems remain, such as determining whether a polynomial-time algorithm exists to decide whether $H(F) = 1$.

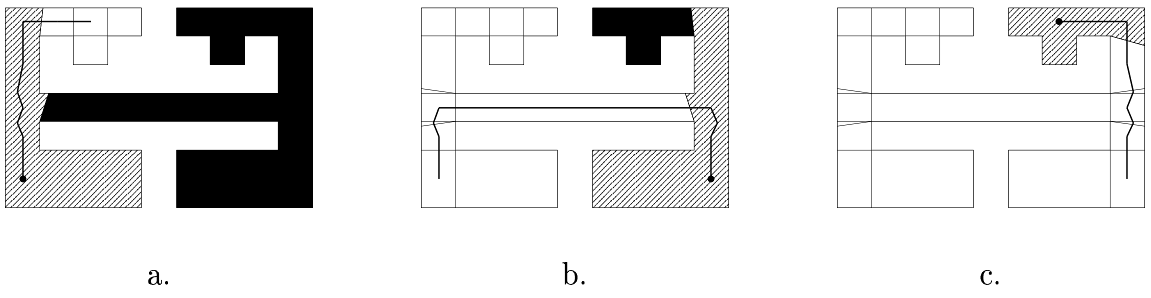

Figure 12: A computed solution trajectory is shown in three frames. The black area represents the contaminated region, and the white area represents the cleared region. The thick curve shows a portion of computed trajectory, which is continued in each frame. The shaded region indicates the visibility region at the final time step of the indicated portion of the trajectory. The thin lines in the cleared region indicate the cell boundaries. In the final snapshot, there is no place remaining where the evader could be hiding.

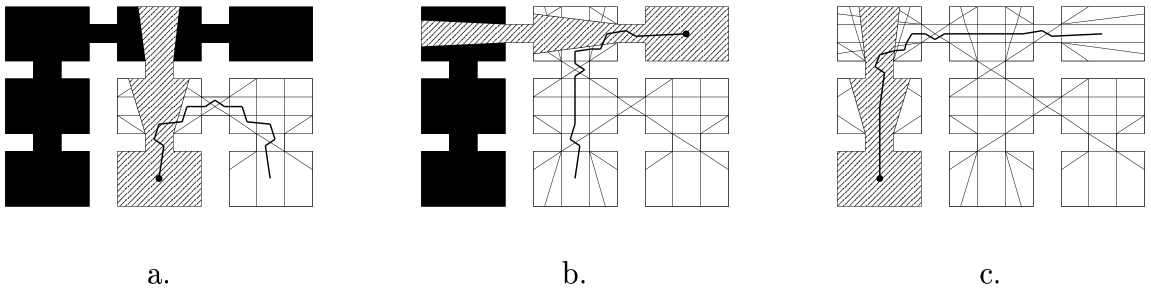

Figure 13: Another computed example.

Information space concepts were used to provide a natural characterization of the unique problem states. The visibility-based pursuit-evasion problem was established as NP-hard. The general concept of partitioning the information space on the basis of critical information changes was introduced to develop a complete algorithm. For the case in which $H(F) = 1$, the algorithm was implemented, and several examples were shown that were computed in a few seconds or less on a standard workstation. Considerable issues remain for the case in which $H(F) = 2$, and in general, approximation algorithms might provide the only hope of obtaining practical solutions to many problems.

Several variations and extensions of the problem are worth exploring. For a variety of

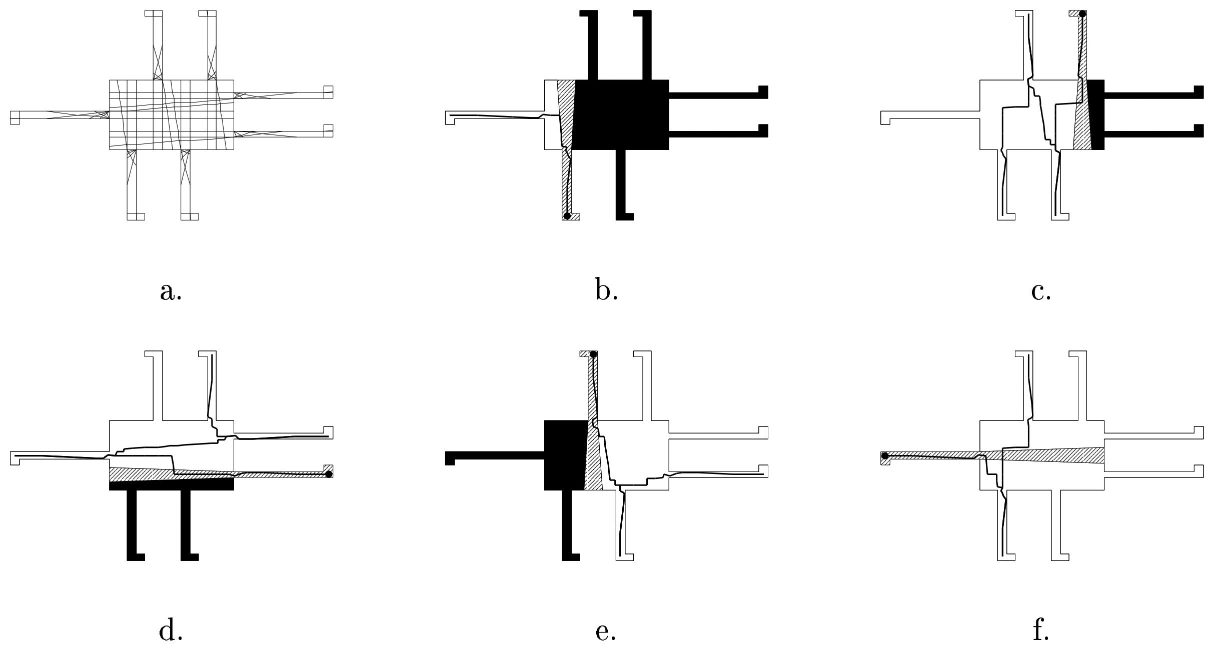

Figure 14: This difficult example requires two recontaminations of the leftmost corridor.

problems, such as visual searching using mobile robots, one can assume initial positions for the pursuers. In addition to a visibility region, each pursuer could have a region of capture, and the task could be to capture the evader using one or more pursuers. The problem can be made more challenging by strengthening the model to include a bounded velocity, or possibly stochastic prediction. The topological issues could become significantly more complex for 3-D free spaces. The conservative cell and edge-visibility concepts could be applied for the 3-D case, but considerable challenges would be faced to produce an efficient algorithm. Another problem variation is to consider a limited viewing angle, or a set of viewing rays as considered in [23]. A limited viewing angle can realistically occur in applications, and the problem can be extended to planning strategies that sweep viewing angles in addition to moving the pursuers. Finally, a cost functional could be defined, leading to problems such as finding the evader in minimum time.

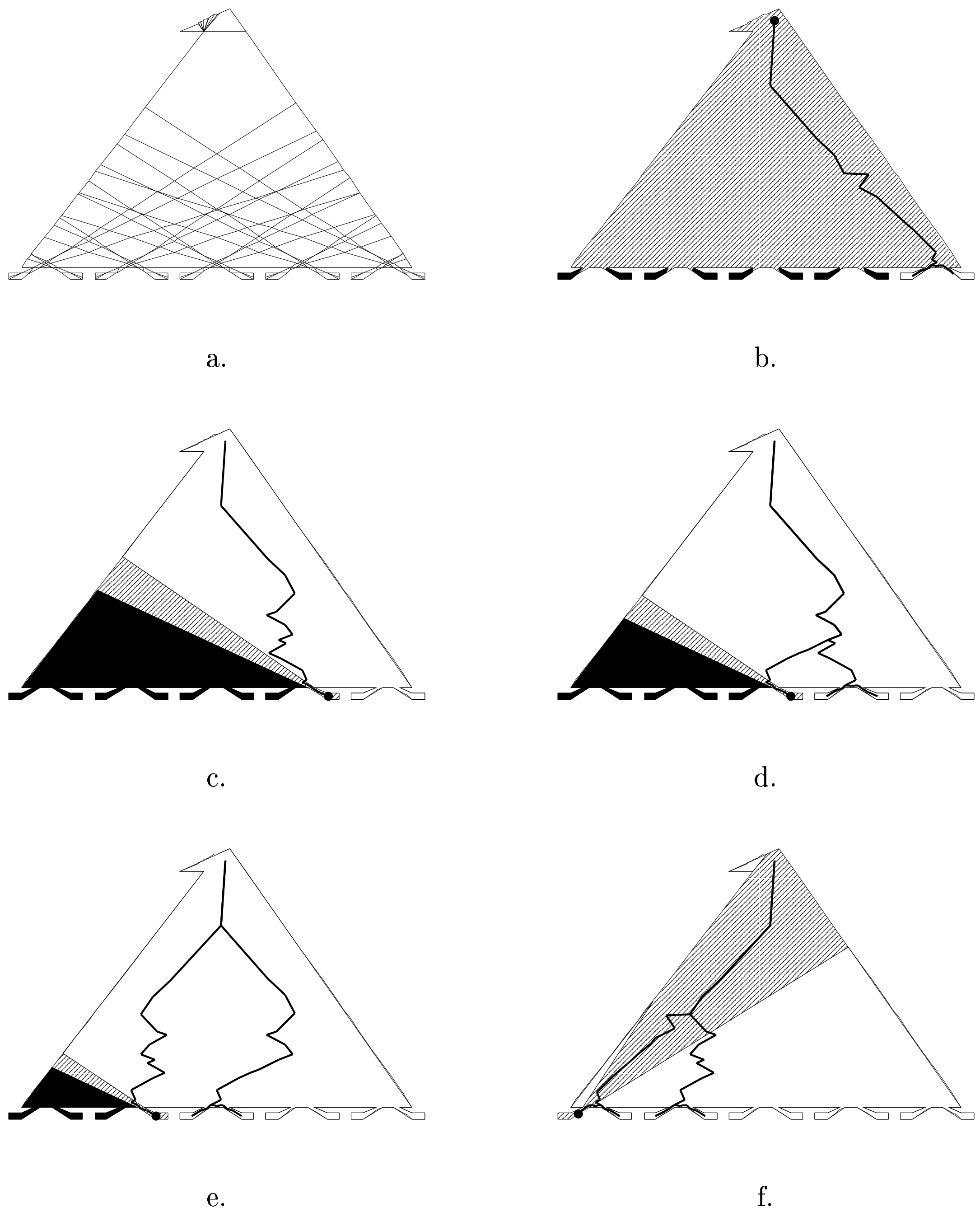

Figure 15: This example requires three recontaminations, and represents one in the sequence that requires a linear number of recontaminations.

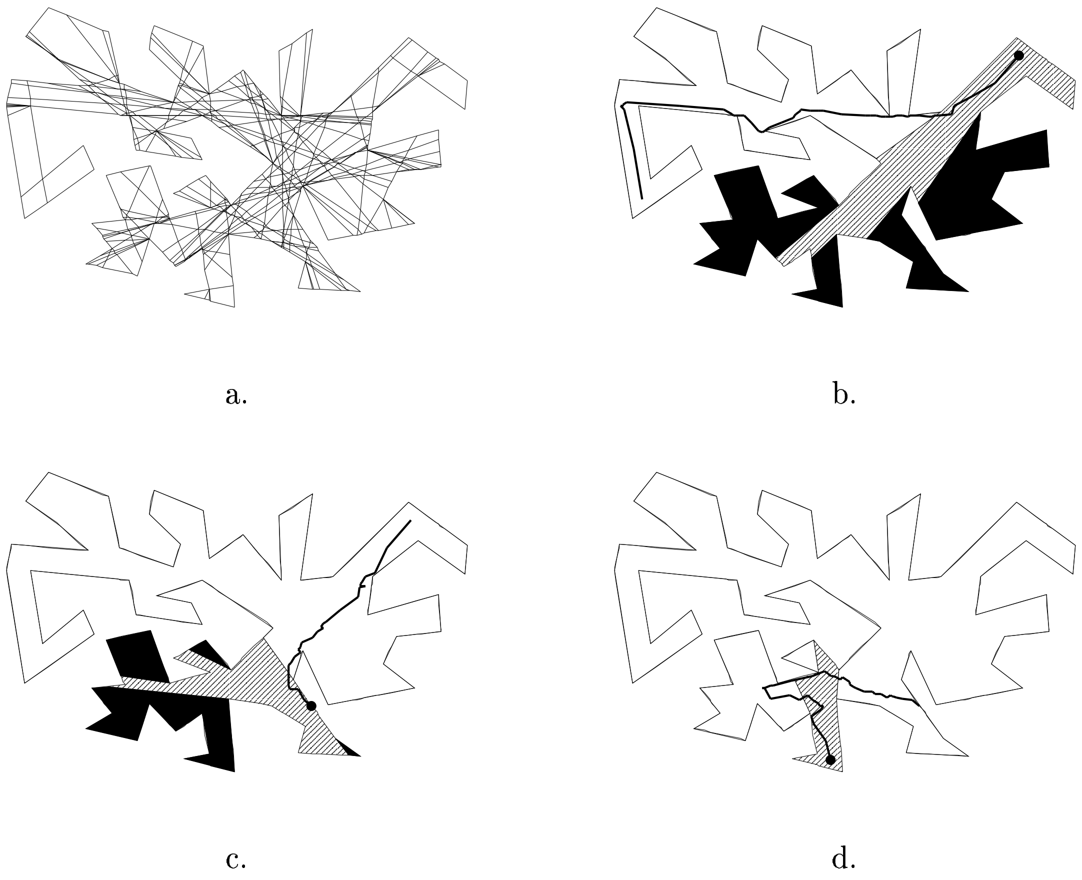

Figure 16: This bad example yields many edge-visibility cells.

Acknowledgments

The research of Leonidas J. Guibas is supported by NSF grant CCR-9623851 and US Army MURI grant 5-23542-A. Rajeev Motwani's research is supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Partnership Award, an ARO MURI Grant DAAH04-96-1-0007, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation. The remaining researchers are supported by ARO MURI grant DAAH04-96-1-007 and ONR grant N00014-94-1-0721. The authors thank Julien Basch, Frédéric Cazals, Bruce Donald, Héctor González-Baños, Gary Kalmanovich, Jon Kleinberg, and Li Zhang, for their helpful suggestions. Gary Kalmanovich suggested the linear recontamination example from Figure 5.

References

[1] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.

[2] D. Bienstock and P. Seymour. Monotonicity in graph searching. *J. Algorithms*, 12:239–245, 1991.

[3] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[4] B. Chazelle. A theorem on polygon cutting with applications. In *Proc. 23rd Annu. IEEE Sympos. Found. Comput. Sci.*, pages 339–349, 1982.

[5] W.-P. Chin and S. Ntafos. Optimum watchman routes. *Information Processing Letters*, 28:39–44, 1988.

[6] D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor – the open edge variant of the polygon search problem. *Int. J. Comput. Geom. & Appl.*, 5(4):397–412, 1995.

[7] M. Erdmann. Randomization for robot tasks: Using dynamic programming in the space of knowledge states. *Algorithmica*, 10:248–291, 1993.

[8] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *AMC Trans. Graphics*, 4(2):74–123, 1985.

[9] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Proc. 1st Workshop on Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.

[10] O. Hájek. *Pursuit Games.* Academic Press, New York, 1975.

[11] R. Isaacs. *Differential Games.* Wiley, New York, NY, 1965.

[12] A. S. Lapaugh. Recontamination does not help to search a graph. *J. ACM*, 40(2):224–245, April 1993.

[13] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.

[14] S. M. LaValle, D. Lin, L. J. Guibas, J.-C. Latombe, and R. Motwani. Finding an unpredictable target in a workspace with obstacles. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 737–742, 1997.

[15] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36:177–189, 1979.

[16] F. Makedon and I. H. Sudborough. Minimizing width in linear layouts. In *Proc. 10th ICALP, Lecture Notes in Computer Science 154*, pages 478–490. Springer-Verlag, 1983.

[17] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35(1):18–44, January 1988.

[18] B. Monien and I. H. Sudborough. Min cut is NP-complete for edge weighted graphs. *Theoretical Computer Science*, 58:209–229, 1988.

[19] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, New York, NY, 1987.

[20] T. D. Parsons. Pursuit-evasion in a graph. In Y. Alani and D. R. Lick, editors, *Theory and Applcation of Graphs*, pages 426–441. Springer-Verlag, Berlin, 1976.

[21] J. T. Schwartz and M. Sharir. On the piano movers' problem: III. Coordinating the motion of several independent bodies. *Int. J. Robot. Res.*, 2(3):97–140, 1983.

[22] T. Shermer. Recent results in art galleries. *Proc. IEEE*, 80(9):1384–1399, September 1992.

[23] I. Suzuki and M. Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM J. Comput.*, 21(5):863–888, October 1992.

[24] R. Talluri and J. K. Aggarwal. Mobile robot self-location using model-image feature correspondence. *IEEE Trans. Robot. & Autom.*, 12(1):63–77, February 1996.

[25] M. Yamashita, H. Unemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers. Technical Report TR-96-07-01, Dept. of Electrical Engineering and Computer Science, University of Wisconsin - Milwaukee, July 1996.