An Algorithm for Searching a Polygonal Region with a Flashlight

Steven M. LaValle Borislav H. Simov Giora Slutzki
Dept. of Computer Science
Iowa State University
Ames, IA 50011 USA
{lavalle, simov, slutzki}@cs.iastate.edu

ABSTRACT

We present an algorithm for a single pursuer with one flashlight searching for an unpredictable, moving target in a 2D environment. For a simple polygon with n edges, the algorithm uses $O(n^2)$ time to decide whether the polygon can be cleared by a 1-searcher, and if so, constructs a search schedule. The key ideas in this algorithm include a representation called the visibility obstruction diagram and a decomposition of this diagram based on a skeleton that arises from critical visibility events. An implementation is presented along with a computed example.

1. INTRODUCTION

Consider the following scenario. In a (dark, doorless) polygonal region there are two moving agents (represented as points). The first one, called the **pursuer**, has the task to find the second one, called the **evader**. The evader can move arbitrarily fast, and his movements are unpredictable by the pursuer. The pursuer is equipped with a flashlight and can see the evader only along the illuminated line segment it emits. The pursuer (a.k.a. 1-searcher) **wins** if she illuminates the evader with her flashlight or if both happen to occupy the same point of the polygon. Clearly, the pursuer should, at all times, be located on the boundary of the polygon and use the flashlight as a moving boundary between the portion of the polygon that has been **cleared** (i.e., the evader is known not to hide there) and the **contaminated** portion of the polygon (i.e., the part in which the evader might be hiding). If there is a movement strategy of the pursuer whereby she wins regardless of the strategy employed by the evader, we say that the polygon is **searchable with one flashlight**, or **1-searchable**.

The problem above was introduced by Suzuki and Yamashita [11]. The main points of interest are the existence and complexity of an algorithm which, given a simple polygon P with n edges, decides whether P is 1-searchable and if so, outputs a search schedule. Although the problem has been open for a while, no complete characterizations or efficient algorithms were developed. Naturally, several, restricted variants were considered. Previously, Icking and Klein [7] had defined the two-guard walkability problem, which is a search problem for two guards whose starting and goal position are given, and who move on the boundary of a polygon so that they are always mutually visible. Icking and Klein gave an $O(n \log n)$ solution, which later was improved by Heffernan [5] to the optimal $\Theta(n)$. Tseng et al [12] solved the two-guard walkability problem in which the starting and goal positions are not given. Recently, Lee et al [9] defined 1-searchability for a room (i.e., a polygon with one door — a point which has to remain clear at all times) and presented an $O(n^2)$ solution. In this paper we solve the original problem defined in [11], and we show that it is a nontrivial generalization of the variants of 1-searchability defined in [7] and [9].

Originally, the problem of 1-searchability of a polygon was introduced together with a more general problem in which the pursuer has 360° vision [11]. For results concerning 360° vision refer to [11, 2, 4, 10] for search in polygons and to [8] for curved planar environments.

Our models are motivated in part by the desire in mobile robotics systems to develop simple sensing mechanisms and to minimize localization requirements (knowing the precise location of the robot). The "flashlight" could be implemented by a camera and vision system that uses feature detection to recognize a target. Alternatively, a single laser beam could be used to detect unidentified changes in distance measurements. Many localization difficulties are avoided since the robot is required to follow the boundary of the environment. Sensors could even be mounted along tracks that are fastened to the walls of a building, as opposed to employing a general-purpose mobile robot. Although it is obviously restrictive to consider only environments that can be cleared by a single pursuer, the problem considered in this paper is surprisingly challenging. It may be possible to extend some of our ideas to allow the coordination of multiple pursuers, eventually broadening the scope of applications.

The rest of the paper is organized as follows. Section 2 introduces the notation and provides observations which reduce pursuit-evasion by a 1-searcher to a search problem in $S^1 \times S^1$, i.e., in a torus. In Section 3.1 we define critical

Proceedings of *16 ACM Symposium on Computational Geometry (SCG'00)*
June 12 - June 14, 2000, Hong Kong

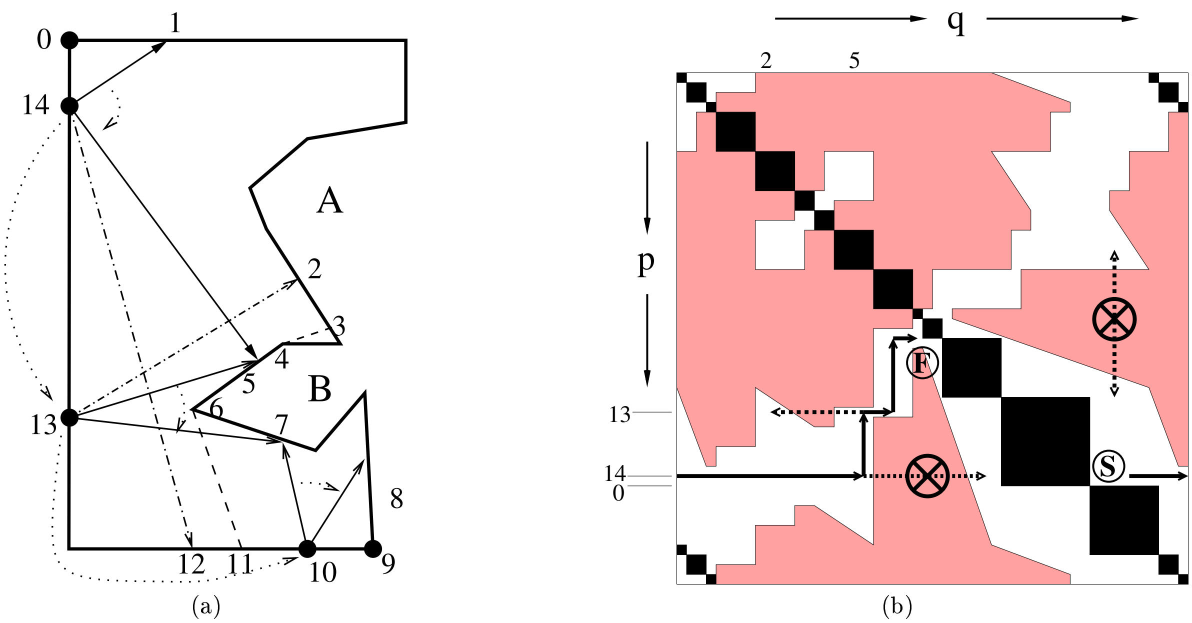

Figure 1: (a) a simple polygon (b) corresponding visibility obstruction diagram

points on the boundary which are essential for determining the schedule of the 1-searcher. These points help us in Section 3.2 to simplify the search from Section 2 to a search in a finite maze. In Section 3.3 we present an $O(n^2)$ algorithm which, given a polygon with n edges, decides whether the polygon is 1-searchable and if so, outputs a schedule for the pursuer. Section 4 discusses an implementation of the algorithm, and also the relationship between our result and the results in [7, 5, 12], [3] and [9]. Section 5 concludes the paper with a summary and directions for future research.

2. NOTATION AND PRELIMINARIES

2.1 Notation

Let P be a simple polygon. (From now on, a polygon is always assumed to be simple.) We denote the **boundary** of P by ∂P. We assume that $\partial P \subseteq P$ and that ∂P is oriented in the **clockwise** (also called **positive**) direction. For any two points $a, c \in \partial P$, we write (a, c) to denote the set of all points $b \in \partial P$ such that when starting after a in positive direction along ∂P, b is reached before c. We also use the notation $[a, c]$, $[a, c)$ and $(a, c]$ for the closed and half-closed intervals on ∂P.

Let $p_0, p_1, \ldots, p_{n-1}$ denote the **vertices** on P ordered in the positive direction. The **edges** of ∂P are $e_0, e_1, \ldots, e_{n-1}$, where edge e_i has endpoints p_i and p_{i+1}, where $i \in \mathbb{Z}_n$ (i.e., the indices are computed modulo n; e.g., $p_0 = p_n$).

We use the standard definition of visibility. For points $c, d \in \partial P$ we say that d is **visible** from c, if every interior point of the line segment $\overline{cd}$ lies in $P - \partial P$. Obviously, if one point is visible from another, then the two are mutually visible. Note that any two points on the same edge of P are not mutually visible.

2.2 Rules of pursuit

We start with a simple example of how the pursuer can clear the polygon in Figure 1(a). Initially, she is at point 0 with the flashlight pointing at 0. To start the search she moves from point 0 to point 14, while at the same time[1] she rotates the flashlight from point 0 to point 1. Next, the pursuer, while staying at 14, rotates the flashlight from point 1 clockwise to point 5. Then she moves from point 14 to point 13 constantly illuminating point 5. Following that, she rotates the flashlight from point 5 to point 7, and then moves from point 13 to point 10. After a final rotation of the flashlight from point 7 to point 8, the pursuer is at point 10 illuminating point 8. The search is complete when she moves from 10 to 9 and simultaneously rotates[1] the endpoint of the flashlight from 8 to 9.

The next observations follow immediately. We can assume that the evader moves only along the boundary. As we mentioned above, the pursuer also moves along ∂P and uses the flashlight to separate the clear and contaminated portions of the polygon. Furthermore, without loss of generality, we will assume in the rest of the paper that the clear portion of the polygon is always to the left of the pursuer as she looks in the direction of the beam of light. We call this assumption the **left invariant**. Thus, a single pair of points is sufficient to record the current status of the pursuit as seen by the pursuer. We define a **configuration** to be a pair $\langle p, q \rangle$ of points $p, q \in \partial P$, and the space of all configurations X to be:

$$X = \{\langle p, q \rangle \mid p, q \in \partial P\} .$$

Let $X_d \subset X$ denote the set of all diagonal configurations $\langle p, q \rangle$ such that p and q lie on the same edge of ∂P. We also denote by $X_v \subseteq X$ all configurations of mutually visible

[1]Note that the beginning and the end of the search are the only times when the pursuer has to simultaneously move herself and rotate the flashlight. Without loss of generality, we can assume that the rest of the time she alternates between pursuer moves and flashlight rotations.

points, i.e., pairs $\langle p,q\rangle$ such that q is visible from p. Intuitively, for $\langle p,q\rangle \in X_v$, p represents the position of the pursuer, while q represents the point illuminated by the flashlight. From the left invariance assumption it follows that all points between p and q (along ∂P) are clear and all the points between q and p are contaminated. Finally, let $X_n = X - X_d - X_v$.

Definition 2.1. *The* ***visibility obstruction diagram*** *or* ***VOD*** *for a polygon P is defined as the triple $\langle X_d, X_v, X_n\rangle$, where X_d, X_v and X_n are defined as above.*

Figure 1 provides an example of a simple polygon and the VOD corresponding to it. The black squares along the diagonal of Figure 1(b) represent X_d, the shaded area represents X_n, and the white area represents X_v. The solution described in Figure 1(a) is plotted as a directed rectilinear path (connected sequence of directed segments in X, shown as solid arrows in Figure 1(b)) starting from the letter 'S' in the lower right corner, wrapping around and ending at the letter 'F'. Each of the motions of the pursuer and the rotations of the flashlight can be represented by vertical or horizontal directed segments in X as follows:

1. **Flashlight rotation in clockwise direction over visible points** and stationary pursuer corresponds to a horizontal segment in X_v directed from left to right. In Figure 1, the rotation from point 1 to point 5 in the beginning of the solution is represented by the first (horizontal) arrow in the path.
2. **Flashlight rotation in counterclockwise direction over visible points** and stationary pursuer corresponds to a horizontal segment in X_v directed from right to left.
3. **Motion in counterclockwise direction** of the pursuer with the flashlight illuminating the same point corresponds to a vertical segment in X_v directed up. In Figure 1, the move of the pursuer from point 14 to point 13 while illuminating point 5 is represented by the second (vertical) arrow in the path.
4. **Motion in clockwise direction** of the pursuer with the flashlight illuminating the same point corresponds to a vertical segment in X_v directed down.
5. **Recontamination move** in which a stationary pursuer moves the flashlight to the left across an interval of invisible points corresponds to a horizontal segment directed from right to left over an area of X_n. In Figure 1, if the pursuer is in point 13 and rotates the flashlight from point 5 to point 2 (over the invisible points between 3 and 4), the move is represented as the right-to-left dashed-dotted arrow in X.

All five moves preserve the left invariant, and each of the first four moves is reversible, i.e., performing a move and its inverse preserves the left invariant. On the other hand, the recontamination move is not reversible. In fact, if the pursuer tries to rotate the flashlight clockwise across an invisible interval this will invalidate the left invariant. For example, in Figure 1(a), if the pursuer at point 14 rotates the flashlight from point 5 to point 12 (over the invisible points between 6 and 11), this will cause the whole polygon to be recontaminated and hence will invalidate the left invariant. This invalid move is shown as the left-to-right dotted arrow in Figure 1(b) . Note that, technically, recontamination happens every time when the pursuer rotates the flashlight in counterclockwise direction. However, since counterclockwise rotation over visible points only is a reversible move, we consider it as a weak form of recontamination. We reserve the term "recontamination" for the irreversible counterclockwise rotation of the flashlight over an interval of nonvisible points.

The following vertical or horizontal directed segments in X do not have corresponding moves of the pursuer:

- segment from left to right which crosses X_n: as explained above, it does not preserve the left invariant.
- vertical segment which crosses X_n: this would imply that either the pursuer leaves ∂P, or for some configuration, the endpoint of the flashlight is not visible by the pursuer.
- directed segment which crosses X_d: this corresponds to the pursuer pointing the flashlight outside the polygon.

Definition 2.2. *Let P be a simple polygon with a configuration space X. A* ***legal path*** *is a continuous rectilinear path in X which uses only the five valid moves described above. A legal path which starts immediately above X_d and ends immediately below X_d is a* ***winning legal path****.*

We use the phrase "immediately above (below)" in the definition in order to allow the pursuer to start (end) the search with a continuous motion of herself and the flashlight. Also note that the circularity of ∂P implies that there is a vertical and horizontal wraparound along each of the axes.

If for some polygon P there exists a winning legal path which does not contain a recontamination move, we say that P **can be cleared without recontamination**. On the other hand, if all of the legal winning paths for P contain a recontamination move, we say that **clearing P requires recontamination**. (For a more detailed example of a polygon, which requires recontamination, see Figure 5, discussed in detail in Section 4.)

3. ALGORITHM FOR FINDING A SCHEDULE FOR THE 1-SEARCHER

In this section we introduce an algorithm which given a polygon outputs either a description of a winning legal path, or a message that the polygon is not 1-searchable. In Section 3.1, using simple geometric conditions, we define points which are of crucial importance for finding a schedule for the 1-searcher. We call them "red points" and later we show that they provide basis for the alternating order of the moves of the pursuer and the rotations of the flashlight. In Section 3.2 we use the red points to construct a more compact "skeleton" representation of the VOD. Finally, in Section 3.3 we show how to reduce the search for a winning legal path in X to a breadth-first search in the "skeleton" space.

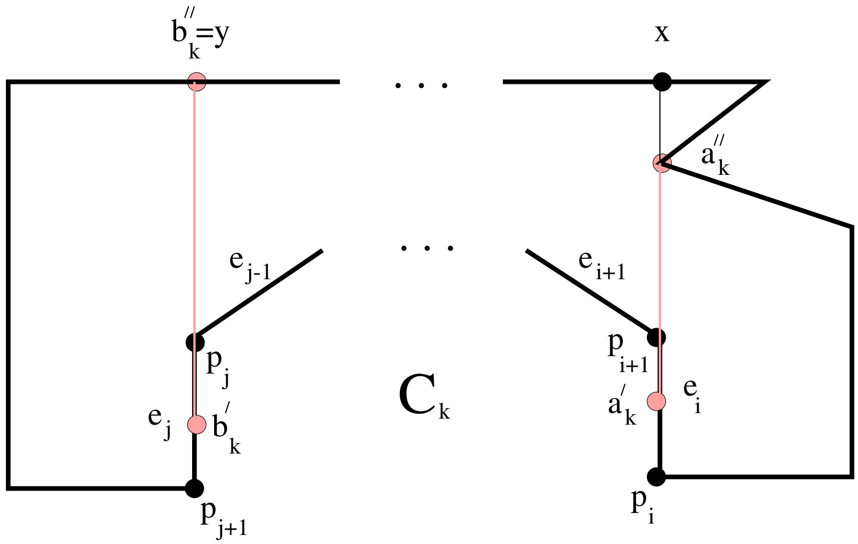

Figure 2: Example of left and right red edges

3.1 Red points

Before we proceed with the description of the algorithm, we need a few more definitions. Vertex $p_i \in \partial P$ is a **reflex vertex** if the angle formed by edges e_{i-1} and e_i, in the interior of P, is greater that 180° (i.e., points p_{i-1}, p_i, and p_{i+1}, form a left turn). Otherwise, p_i is a **non-reflex vertex**. A maximal subinterval of ∂P of the form $C = (p_i, p_{j+1})$, where all the vertices $p_{i+1}, \ldots, p_j$ are reflex vertices, forms a **concave region**. Obviously, two concave regions cannot overlap and must be separated by non-concave regions (each of which contains at least one non-reflex vertex). For example, Figure 1(a) shows two concave regions labeled A and B.

Consider the k-th concave region $C_k = (p_i, p_{j+1})$ of P, see Figure 2. Define a'_k and b'_k to be the midpoints of the edges e_i and e_j, respectively. Shoot a ray [1, 6] starting at point a'_k through p_{i+1} and let x be the point of ∂P where the ray leaves P for the first time. Define a''_k to be the point in $\overline{a'_k x} \cap \partial P$ which is farthest from a'_k in positive direction along the boundary. Similarly, shoot a ray starting at point b'_k through p_j and let y be the point of ∂P where the ray leaves P for the first time. Define b''_k to be the point in $\overline{b'_k y} \cap \partial P$ which is farthest from b'_k in negative direction along the boundary.

Given the points a'_k and a''_k define $\mathcal{H}(a'_k) = (a'_k, a''_k]$. Note that a'_k is not visible from any point in $\mathcal{H}(a'_k)$. Thus, intuitively, whenever the pursuer stays in $\mathcal{H}(a'_k)$, the evader can hide in a'_k and is guaranteed that he will not be detected. Similarly, we can define $\mathcal{H}(b'_k) = [b''_k, b'_k)$.

We have defined, for each concave region C_k of P, four types of important points a'_k, b'_k, a''_k and b''_k; these are called the **red points** of P and the set of all such points, for the various concave regions of P is denoted by A', B', A'', and B'' respectively.

Lemma 3.1. *Let P be a polygon with n edges. The red points of P and their order along ∂P can be found in time $O(n \log n)$.*

Proof: The points in A' and B' can be found in time $O(n)$. To find the points in A'' and B'', we can use the ray-shooting algorithm of Chazelle et al [1] with a total time of $O(n \log n)$. ∎

3.2 Skeletons

Instead of trying to determine the exact shape of X_v, in this section we will use the information obtained from the red points to define an "equivalent" search space that is computationally much more convenient.

Definition 3.2. *Let P be a polygon with m concave regions and a'_k, a''_k, b'_k and b''_k be the red points defined for the k-th concave region, $0 \le k \le m-1$. We define a **skeleton** $SK \subset X$ to be the union of the following sets of configurations:*

- *a **diagonal** line,* $D = \{\langle p, p\rangle \mid p \in \partial P\}$
- *two unions of **horizontal** lines,* $AH = \bigcup_{k=0}^{m-1}\{\langle a'_k, q\rangle \mid q \in \mathcal{H}(a'_k)\}$ *and* $BH = \bigcup_{k=0}^{m-1}\{\langle b'_k, q\rangle \mid q \in \mathcal{H}(b'_k)\}$
- *two unions of **vertical** lines,* $AV = \bigcup_{k=0}^{m-1}\{\langle p, a'_k\rangle \mid p \in \mathcal{H}(a'_k)\}$ *and* $BV = \bigcup_{k=0}^{m-1}\{\langle p, b'_k\rangle \mid p \in \mathcal{H}(b'_k)\}$

In order to illustrate the intuition behind the term "skeleton", in Figure 3(a) we show X together with SK for the polygon in Figure 1. All the horizontal and vertical lines of SK span from the diagonal to a horizontal or vertical extremum of a X_n region. In a sense, those lines form a supporting frame for the regions of X_n.

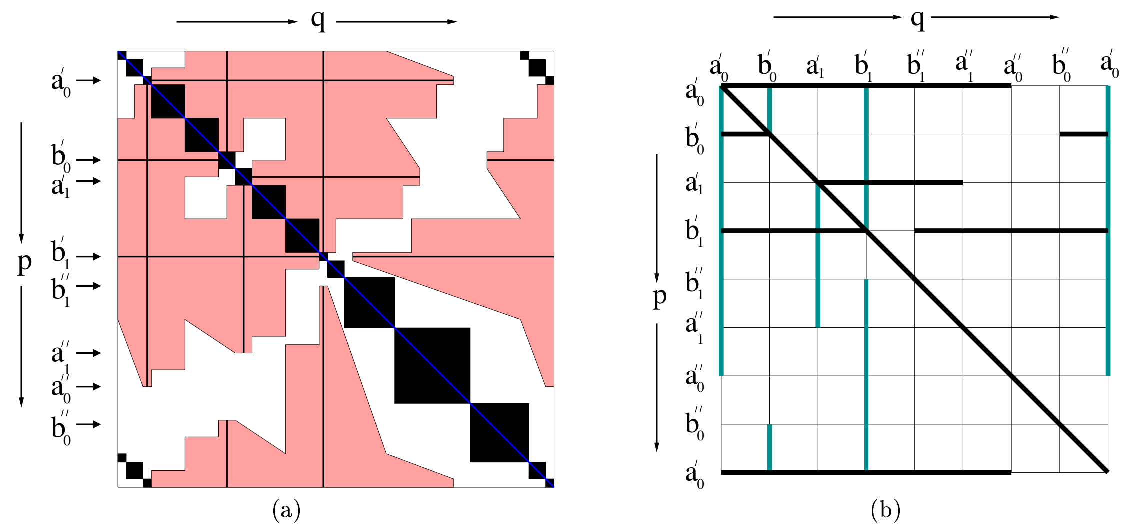

Figure 3: Skeleton (a) and square grid (b) of the polygon from Figure 1

Next we show that if we relax somewhat the restrictions that the regions X_n and X_d impose on a legal path, we can build a simpler representation of X, which will allow us to find a solution in time $O(n^2)$. The solution in the new search space can then be modified into a winning legal path in X using the algorithm in [5]. The idea is to replace the regions X_n and X_d by their skeletons. More formally, define a coloring of the configurations in X as follows:

- all configurations in $D \cup AH \cup BH$, i.e., the diagonal and horizontal lines, are **black**
- all configurations in $SK - (D \cup AH \cup BH)$, i.e., the vertical lines, are **red**
- all configurations in $X - SK$ are **white**

A convenient way to illustrate the coloring of X is shown in Figure 3(b). It represents X over the square grid formed by plotting the red points a'_k, a''_k, b'_k, b''_k, $k \in \mathbb{Z}_m$, along the axes. For convenience we assume that the configuration $\langle a'_0, a'_0 \rangle$ is in the upper left corner of the grid. Note that the thin lines which form the grid are added for reference only and are not part of the skeleton. The thick grey and black lines in the figure correspond to the red and black lines of SK.

Definition 3.3. *A* ***relaxed path*** *in X is a continuous rectilinear path which does not cross a black line and does not cross a red line from left to right. A relaxed path which starts immediately above D and ends immediately below D is a* ***winning relaxed path****.*

The main difference between a relaxed path and a legal path is that the former can enter into X_n in any direction as long as the path does not cross the skeleton. The correctness of our algorithm is based on the following theorem.

Theorem 3.4. *(i) Every winning legal path is a winning relaxed path. (ii) Every winning relaxed path can be transformed into a winning legal path.*

Proof: Since $SK \subset X_n \cup X_d$, the restrictions to a relaxed path are a subset of the restrictions to a legal path, and the correctness of (i) follows immediately. The proof of (ii) is rather lengthy and quite involved. An outline of the argument is presented in the appendix. ∎

3.3 Description of the algorithm

The grid is useful to illustrate compactly the restrictions imposed by the skeleton. Intuitively, we can think about a winning relaxed path as a path in a maze. The black lines in X represent rigid walls, while the red lines in X represent one directional doors which can be entered only from right to left. Clearly, we can apply a breadth-first approach to search for a winning relaxed path directly in the grid. However, since breadth-first-search is most often associated with graphs, we will present the search in an equivalent directed graph $\mathcal{G}_P$, representing the connectivity between neighboring squares of the grid. Suppose the sequence $\langle r_0, r_1, \ldots, r_{4m-1} \rangle$ consists of all the red points in A', A'', B', and B'', such that $r_0 = a_0{}'$ and the elements of the sequence are ordered in the positive direction along ∂P. The vertices in the graph correspond to the squares in the grid:

$$V(\mathcal{G}_P) = \{v_{i,j} \mid i, j \in \mathbb{Z}_{4m}\},$$

where vertex $v_{i,j}$ represents the square in the i-th row and j-th column of the grid. To define $E(\mathcal{G}_P)$, the edges in the graph, we first give an intuitive description, which we will follow by a more formal definition.

In order to construct the set $E(\mathcal{G}_P)$, we start with the set E^n of edges between all pairs of vertices corresponding to neighboring squares. From E^n we then subtract E^d, the set of edges across the diagonal, and E^h, the set of edges across a horizontal line. This guarantees that a path in $\mathcal{G}_P$ cannot cross the diagonal or a horizontal line, correspondingly. Finally, we remove E^v, the set of right edges through a vertical line, which ensures that a path in $\mathcal{G}_P$ cannot cross a vertical line from left to right. Formally, the set of edges in $\mathcal{G}_P$ is defined as follows:

$$E(\mathcal{G}_P) = E^n - E^d - E^v - E^h,$$

where

- E^n represents the set of all edges between neighboring squares in the grid:

$$E^n = \{\langle v_{i,j}, v_{i\pm1,j}\rangle, \langle v_{i,j}, v_{i,j\pm1}\rangle \mid i, j \in \mathbb{Z}_{4m}\}$$

- E^d represents the set of all the left and down incoming or outgoing edges from the diagonal squares in the grid:

$$\begin{aligned} E^d \quad = \quad & \{\langle v_{i,i+1}, v_{i,i}\rangle, \langle v_{i,i}, v_{i,i-1}\rangle, \\ & \langle v_{i-1,i}, v_{i,i}\rangle, \langle v_{i,i}, v_{i+1,i}\rangle \mid i \in \mathbb{Z}_{4m}\} \end{aligned}$$

- E^v represents the set of all left-to-right edges between neighboring squares across a vertical red line in the grid

$$\begin{aligned} E^v \quad = \quad & \{\langle v_{i,j-1}, v_{i,j}\rangle \mid \\ & \exists k : ((r_j = a'_k) \wedge (r_i \in [a'_k, a''_k)) \vee \\ & ((r_j = b'_k) \wedge (r_i \in [b''_k, b'_k))\} \end{aligned}$$

- E^h represents the set of all edges between neighboring squares across a horizontal black line in the grid

$$\begin{aligned} E^h \quad = \quad & \{\langle v_{i-1,j}, v_{i,j}\rangle, \langle v_{i,j}, v_{i-1,j}\rangle \mid \\ & \exists k : ((r_i = a'_k) \wedge (r_j \in [a'_k, a''_k)) \vee \\ & ((r_i = b'_k) \wedge (r_j \in [b''_k, b'_k))\} \end{aligned}$$

Finding a winning relaxed solution is equivalent to finding a nontrivial (nonzero length) path in the graph $\mathcal{G}_P$ starting from some diagonal vertex $v_{i,i}$ and ending at some diagonal vertex $v_{j,j}$. Finding a path in $\mathcal{G}_P$ can be done using breadth-first search, and it takes time linear in the size of the graph. Note that the search in $\mathcal{G}_P$ will find a relaxed, but not necessarily legal, path π. However, according to Theorem 3.4, there exists a corresponding winning legal path p. We can find p by breaking up π into monotonous[2] subpaths (possibly interrupted by recontamination) and applying the algorithm of Heffernan [5] for constructing straight or counter walks, successively on each part of the polygon between the subpaths.

THEOREM 3.5. *There is an algorithm that, given a simple polygon P with n edges, in time $O(n^2)$ decides whether P can be cleared by a 1-searcher, and if so, outputs a search schedule.*

[2]A subpath in X is monotonous if the corresponding moves of the pursuer and the flashlight on ∂P are monotonous. A monotonous path in X corresponds to either a straight or a counter walk as defined in [5]

```
  FIND_RELAXED_SOLUTION()
1 input a polygon as a sequence of points ⟨p0, p1, ... pn−1⟩
2 determine the points {a'0, ... a'm−1} and {b'0, ... b'm−1}
3 use ray shooting to find points a''i, b''i for i ∈ ℤm
4 sort the red points into a sequence ⟨r0, r1, ... , r4m−1⟩
5 construct the graph 𝒢P
6 using BFS in 𝒢P find a path from vi,i to vj,j
7 output the path or a message if one does not exist
```

Figure 4: Outline of the algorithm for finding a winning relaxed path

Proof: An outline of an algorithm which, given a simple polygon, finds a search schedule for an 1-searcher is presented in Figure 4. The correctness of the algorithm follows immediately from Theorem 3.4 and the definition of the graph $\mathcal{G}_P$.

We have to show that the algorithm runs in time $O(n^2)$. Steps (1) and (2) of the algorithm take time $\Theta(n)$. Note that the number of concave regions, m, cannot exceed the number of edges in the polygon. Thus the number of red points is exactly $4m$ which is $O(n)$. From Lemma 3.1 it follows that step (3) can be done in time $O(n \log n)$. Step (4) takes time $O(m \log m)$. Steps (5) and (6) can be completed in time $O(m^2)$ since the graph has $O(m^2)$ vertices and the outdegree of each vertex is bounded by 4.

Thus the total running time is $O(n \log n + m^2)$, which is equivalent to $O(n^2)$ in the worst case, when $m = \Theta(n)$. ∎

4. IMPLEMENTATION AND COMPARISON WITH PREVIOUS WORK

The algorithm outlined in Figure 4 was implemented using GNU C++ and the Library of Efficient Data Types and Algorithms (LEDA), and experiments were performed on a Pentium III 500Mhz PC running Linux. The algorithm was determined to be efficient enough for practical use in real environments. Figure 5 provides an example of the program output. Note that the position of the pursuer on the boundary is designated with a small white circle.

The polygon in Figure 5 is interesting beyond a mere illustration of the algorithm implementation. It represents a polygon which **requires** a recontamination in order to be searched successfully. The recontamination happens between frames (d) and (e): the endpoint of the flashlight **jumps** over an interval of ∂P, thus some of the area which was already cleared is contaminated again after the pursuer rotates the beam to the left.

It is important to note that a schedule for a 1-searcher which contains a recontamination move cannot be simulated by a corresponding search schedule with two guards [7, 5, 12]. The reason is that the two guards cannot maintain visibility and stay on ∂P while attempting to simulate a jump of the flashlight. Therefore, there are polygons that can be cleared by a 1-searcher but cannot be cleared by two guards. Since every schedule for two guards is a schedule for a 1-searcher as well, it follows that the set of polygons that can be cleared by a 1-searcher is a strict superset of the polygons that can be cleared by two guards. Also, note that while the polygon

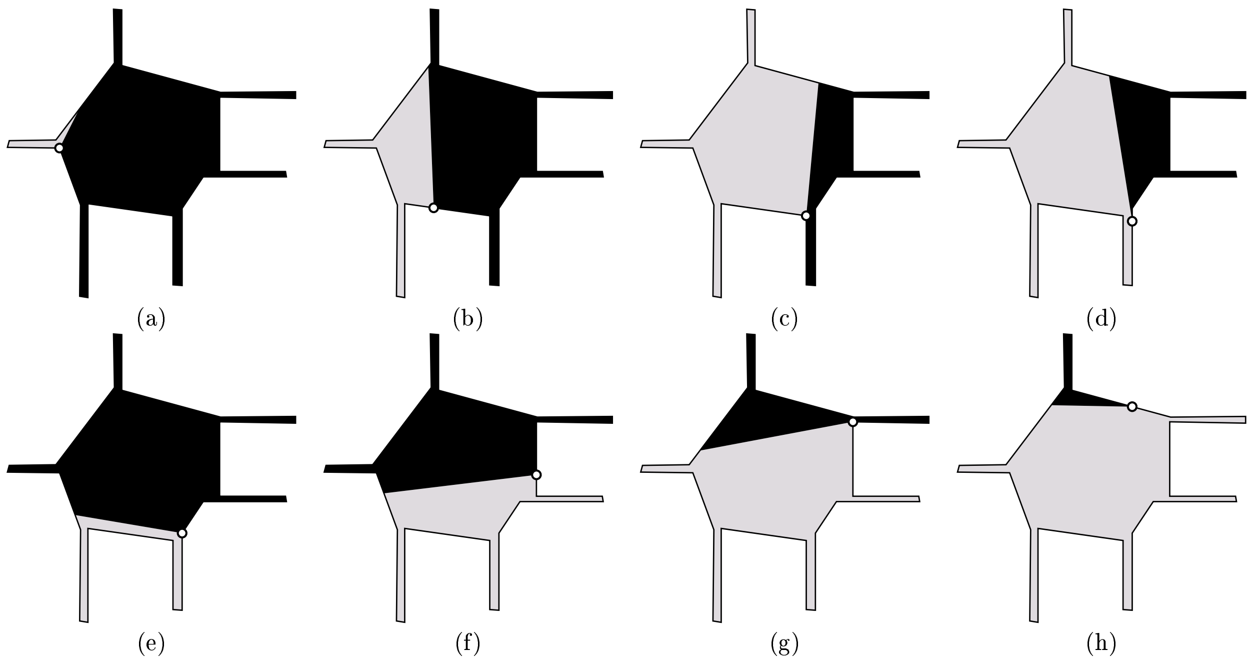

Figure 5: A polygon which requires recontamination (between frames (d) and (e)).

in Figure 5 can be cleared by a chain of three guards using an algorithm by Efrat et al [3] (this is generalization of the two guards problem to a chain of k guards), this is not equivalent to finding a solution for a 1-searcher.

Similarly, it is not hard to show that every point in the polygon in Figure 5 is contaminated at some time during any successful 1-searcher schedule. Thus there is no point d such that the room $\langle P, d\rangle$ can be cleared by a 1-searcher as described in the $O(n^2)$ algorithm of Lee et al [9].

5. CONCLUSION

In this paper we have presented an $O(n^2)$ algorithm which, given a simple polygon with n edges, decides whether the polygon can be cleared by a 1-searcher and if so, outputs a search schedule. The algorithm is a nontrivial generalization of the two-guard search algorithm and solves a (rather longstanding) problem left open in [11]. The most interesting extension of the result in the current paper would be an algorithm which, given a polygon and an integer k, decides whether the polygon can be cleared by k 1-searchers, and ideally, returns a search schedule. Note that the problem does not impose any restrictions on the mutual visibility between the 1-searchers which, we can easily show, allows clearing of a strictly greater set of polygons as compared to a chain of $k+1$ guards. Finally, while the proposed problem for k 1-searchers is NP-hard for polygonal regions which contain holes [13], little is known about the complexity of the problem for simple polygons.

Acknowledgments

We are grateful for the funding provided in part by NSF CAREER Award IRI-9875304 (LaValle).

REFERENCES

[1] B. Chazelle, H. Edelsbrunner, M. Grigni, L. Guibas, J. Hershberger, M. Sharir, and J. Snoeyink. Ray shooting in polygons using geodesic triangulations. *Algorithmica*, 12:54–68, 1994.

[2] D. Crass, I. Suzuki, and M. Yamashita. Searching for a mobile intruder in a corridor — the open edge variant of the polygon search problem. *International Journal of Computational Geometry and Applications*, 5(4):397–412, 1995.

[3] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali. Sweeping simple polygons with a chain of guards. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2000. to appear.

[4] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani. Visibility-based pursuit-evasion in a polygonal environment. In F. Dehne, A. Rau-Chaplin, J.-R. Sack, and R. Tamassia, editors, *WADS '97 Algorithms and Data Structures (Lecture Notes in Computer Science, 1272)*, pages 17–30. Springer-Verlag, Berlin, 1997.

[5] P. Heffernan. An optimal algorithm for the two-guard problem. *International Journal of Computational Geometry and Applications*, 6:15–44, 1996.

[6] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *Journal of Algorithms*, 18(3):403–431, May 1995.

[7] C. Icking and R. Klein. The two guards problem. *International Journal of Computational Geometry and Applications*, 2(3):257–285, 1992.

[8] S. M. LaValle and J. E. Hinrichsen. Visibility-based pursuit-evasion: The case of curved environments. In *Proceedings, IEEE International Conference on Robotics and Automation (ICRA)*, pages 1677–1682, Detroit, MI, USA, May 1999.

[9] J.-H. Lee, S.-M. Park, and K.-Y. Chwa. Searching a polygonal room with a door by a 1-searcher. *International Journal of Computational Geometry and Applications*, 2000. to appear.

[10] J.-H. Lee, S. Y. Shin, and K.-Y. Chwa. Visibility-based pursuit-evasion in a polygonal room with a door. In *Proceedings, ACM Symposium on Computational Geometry (SCG)*, pages 281–290, Miami Beach, FL, USA, June 1999.

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

[12] L. H. Tseng, P. Heffernan, and D. T. Lee. Two-guard walkability of simple polygons. *International Journal of Computational Geometry and Applications*, 8(1):85–116, Feb. 1998.

[13] M. Yamashita, H. Umemoto, I. Suzuki, and T. Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers. In *Proceedings, ACM Symposium on Computational Geometry (SCG)*, pages 448–450, Nice, France, June 1997.

APPENDIX

A. PROOF OF THEOREM 3.4, PART (II)

In this section we provide an outline of the proof that a winning relaxed path can be transformed into a winning legal path.

LEMMA A.1. *Every horizontal or vertical line of the skeleton has a tip which touches a region in X_v. Specifically, let $\langle a''_k, a'_k\rangle$ be the endpoint of a red line. There exists a sufficiently small rectangle Q with left upper corner $\langle a''_k, u_1\rangle$ and lower right corner $\langle v, u_2\rangle$, where $a'_k \in (u_1, u_2)$ and $a''_k \in (a'_k, v)$, such that the interior of Q lies entirely in X_v. (A similar statement is true for the other three kinds of endpoints of red lines.)*

Proof: The proof follows directly from the definitions of the red points a'_k and a''_k; refer to Section 3.1 and Figure 2. We choose $u_1 = p_i$, $u_2 = p_{i+1}$ to be the endpoints of the edge that a'_k lies on. Also, we choose $v \in \partial P$, such that $v \in (a''_k, a'_k)$, and v is sufficiently close to a''_k such that the edge $\overline{p_i p_{i+1}}$ is visible from every point in the interval (a''_k, v). Clearly, the interior of Q lies in X_v. ∎

LEMMA A.2. *Let $\overline{uv}$ be a horizontal or vertical segment, such that $u, v \in X_v$. If $\overline{uv} \cap X_n \neq \emptyset$, then $\overline{uv} \cap SK \neq \emptyset$.*

Proof: The lemma states the fact that relative to a fixed point p, every maximal interval of points invisible from p has to contain a point $q \in A' \cup B'$, such that $p \in \mathcal{H}(q)$. In other words, if there is a part of the boundary which is not visible from p, it must contain some red point, a'_k or b'_k. ∎

In general, X_v consists of a finite number of maximal connected regions. We call them **conservative regions**, because any path within a region preserves the left invariant[3]. For example, there are three conservative regions in Figure 3(a).

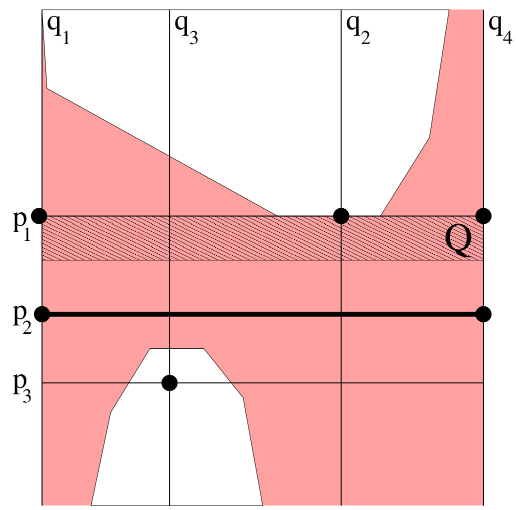

Figure 6: Horizontal separation of conservative regions in the proof of Lemma A.3.

The following lemma states that conservative regions which do not overlap horizontally are separated by black horizontal lines.

LEMMA A.3. *Let C be a conservative region, and let the configuration $\langle p_1, q_2\rangle \in X$ be a local minimum of the boundary of C. For $q_1, q_4 \in \partial P$, such that $q_2 \in (q_1, q_4)$, suppose that the configurations $c_1 = \langle p_1, q_1\rangle$ and $c_2 = \langle p_1, q_4\rangle$ do not lie on the boundary of C. Let Q be a sufficiently small rectangle with an upper edge $\overline{c_1 c_2}$, such that all interior points of Q lie in X_n. For $q_3 \in (q_1, q_4)$, if $\langle p_3, q_3\rangle \in X_v - Q$, then it follows that there is a black horizontal line which separates p_1 and p_3, i.e., there exists $p_2 \in \partial P$, $p_2 \in (p_1, p_3)$ such that the horizontal segment between $\langle p_2, q_1\rangle$ and $\langle p_2, q_4\rangle$ is a part of a black line.*

Proof Sketch: The proof relies on an observation of Guibas et al [4] that when the pursuer moves along the boundary, an interval of visible points appears/disappears exactly when a point on a bitangent line is crossed[4]. Note that in the lemma, the existence of a local minimum $\langle p_1, q_2\rangle$ for the conservative region C means that when a pursuer is moving in the positive direction along the boundary, a visible interval neighboring point q_2 disappears exactly when the pursuer

[3] The conservative regions in our paper have a meaning close to the conservative cells defined by Guibas et al [4].

[4] These are the "green points" defined in [4].

crosses point p_1. Therefore, p_1 must be a point where a bitangent intersects ∂P. Assume that the pursuer continues to move clockwise on the boundary past point p_1. How soon can point q_3 become visible? No point $x \in (q_1, q_4)$, including q_3, is visible before the pursuer crosses another point p_1' of the bitangent. Since there is a point $p_2 \in A' \cup B'$ such that $p_2 \in (p_1, p_1')$ and $(q_1, q_4) \subset \mathcal{H}(p_2)$, it follows that there is a corresponding black horizontal line which includes the segment between the configurations $\langle p_2, q_1\rangle$ and $\langle p_2, q_4\rangle$. ∎

Since describing the exact shape of the conservative regions is rather complicated, we would like to enclose every region in a rectilinear boundary which will be easier to construct and explore.

Definition A.4. *Given a configuration $c \in X$, the set of all configurations reachable from c without crossing the skeleton is called a **visibility tile** (of c).*

Note that every conservative region of X is contained in some visibility tile. If a tile does not contain a conservative region, it is an **empty tile**. Otherwise it is **nonempty**.

Proposition A.5. *Let T be a nonempty tile, let C_0 and C_1 be two conservative regions in T, and c_0 and c_1 be two configurations, such that $c_0 \in C_0$ and $c_1 \in C_1$. If there is a rectilinear path within T from c_0 to c_1 and the path consists of k segments, then there is a rectilinear path from c_0 to c_1 entirely within X_v, i.e., along mutually visible points.*

Proof Sketch: The proof is by induction on k, the number of segments in the rectilinear path between c_0 and c_1.

The basis of the induction includes two cases: $k = 1$ and $k = 2$. The case for $k = 1$ follows immediately from Lemma A.2. The case for $k = 2$ follows from Lemma A.3. The inductive step includes separate arguments for monotonous and nonmonotonous rectilinear paths. (We define a "monotonous rectilinear path" as a rectilinear path, the projection of which on any of the two axes is either a nondecreasing or a nonincreasing vertical/horizontal path.) We omit the details for space considerations. ∎

Corollary A.6. *Each tile T contains at most one conservative region. This implies that, for a nonempty tile T, if $c', c'' \in X_v \cap T$, then there is a path between c' and c'' entirely within $X_v \cap T$.*

Let T_0 and T_1 be two nonempty tiles, and let p, q_0, q_1 be points in ∂P, such that $q_0 \in (p, q_1)$ and the configurations $d_0 = \langle p, q_0\rangle$ and $d_1 = \langle p, q_1\rangle$ are interior points in T_0 and T_1, correspondingly. If the line segment $\overline{d_0 d_1}$ crosses neither the diagonal nor any nonempty tiles other than T_0 and T_1, we say that T_0 is a **left neighbor** of T_1. We assume that neither of the tiles borders the diagonal line. We can make the assumption since our goal with the relation "left neighbor" is to represent the recontamination move of the pursuer, and recontamination from or to a tile which borders the diagonal is not useful. If there is recontamination to a nonempty tile which is immediately above the diagonal, then we can ignore all the previous moves, since this tile is reachable directly from the diagonal. Similarly, if there is recontamination from a nonempty tile which is immediately below the diagonal, we can ignore the rest of the moves, since the diagonal is reachable before the recontamination.

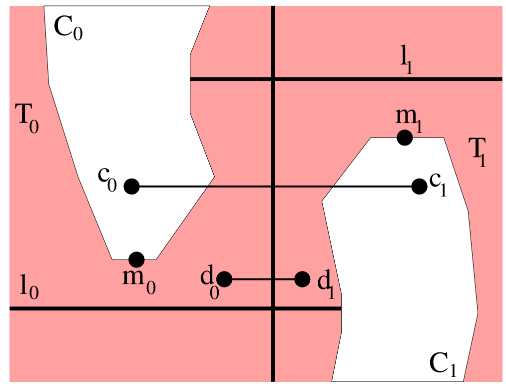

Figure 7: Position of C_0, C_1, T_0, and T_1 in the proof of Lemma A.7.

Lemma A.7. *If T_0 and T_1 are two nonempty tiles such that T_0 is a left neighbor of T_1, then the corresponding conservative regions, C_0 and C_1, overlap horizontally, i.e., there exist $c_0 = \langle p, q_0\rangle \in C_0$ and $c_1 = \langle p, q_1\rangle \in C_1$. Also, the segment $\overline{c_0 c_1}$ does not cross any conservative regions other than C_0 and C_1.*

Proof Sketch: Note that if one of the two tiles T_i is unbounded, then the corresponding conservative region C_i is unbounded and it overlaps horizontally with the whole vertical axis, which proves the claim. Therefore, it suffices to consider the case in which both tiles are bounded from above and below by black lines. Let $L_0 = \langle l_0, \cdot\rangle$ (respectively, $L_1 = \langle l_1, \cdot\rangle$) be the lowest (resp., highest) horizontal black line which borders T_0 (resp., T_1). Similarly, let $m_0 = \langle m_0', m_0''\rangle$ (resp., $m_1 = \langle m_1', m_1''\rangle$) be the global minimum (resp., maximum) on the boundary of the conservative region of T_0 (resp., T_1), see Figure 7.

Since T_0 is a left neighbor of T_1, there exists a horizontal edge $\overline{d_0 d_1}$ which extends from T_0 to T_1. Without loss of generality, assume that $\overline{d_0 d_1}$ lies between lines L_1 and L_0. (The case in which $\overline{d_0 d_1}$ lies between lines L_0 and L_1 is similar, we just have to redefine the lines L_0 and L_1 to the highest and lowest horizontal lines of the corresponding regions.) If we assume that $m_0' \notin (l_1, l_0)$ or $m_0' \notin (l_1, l_0)$, we can apply Lemma A.1 to show that there are nonempty tiles between T_0 and T_1. This would contradict the hypothesis, therefore, it follows that $m_0', m_1' \in (l_1, l_0)$. For the sake of contradiction, assume that the white regions do not overlap horizontally, i.e., $m_0' \in (l_1, m_1')$. We can apply Lemma A.3 by choosing m_0 for the local minimum in the lemma, and any point in X_v sufficiently close to m_1 for the configuration in X_v. It follows that there is a horizontal black line which

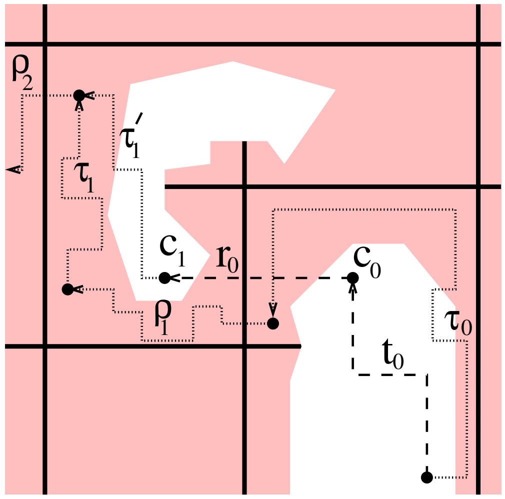

Figure 8: Inductive step in Proposition A.8.

crosses the rectangle with left upper corner m_0 and lower right corner m_1. This implies that the black line lies between the lines L_1 and L_0, so either L_0 is not a bordering line of T_0, or L_1 is not a bordering line of T_1. We have reached a contradiction, therefore our assumption that $m_0' \in (l_1, m_1')$ is false. It follows that C_0 and C_1 overlap horizontally, i.e., there exist $c_0 = \langle p, q_0 \rangle \in C_0$ and $c_1 = \langle p, q_1 \rangle \in C_1$.

Using Lemma A.1 again, we can show that there are no nonempty tiles (other than T_0 and T_1) in the rectangle with left lower corner m_0 and right upper corner m_1. It follows that the horizontal segment $\overline{c_0c_1}$ does not cross any conservative regions other than C_0 and C_1. ∎

Every relaxed path π which starts and ends at a nonempty tile can be partitioned into a sequence of subpaths τ_0, ρ_1, $\tau_1, \ldots, \rho_k$, τ_k such that every subpath τ_i lies completely inside a nonempty tile and every subpath ρ_i starts in the tile of τ_{i-1}, ends in the tile of τ_i, and crosses no other nonempty tiles. On the other hand, a legal path p can be partitioned into a concatenation of subpaths $t_0, r_1, t_1, \ldots, r_k, t_k$ where each t_i is a legal path inside the same conservative region and r_i is a horizontal path which starts from the conservative region of t_{i-1}, ends in the conservative region of t_i, and does not cross other conservative regions. Intuitively, a path t_i corresponds to a sequence of the four reversible moves of the pursuer as defined in section 2.2. A horizontal segment r_i (always right-to-left) corresponds to the fifth, recontamination move: stationary pursuer, rotating the flashlight counterclockwise across an interval of invisible points.

We will show that we can transform every relaxed path π into a legal path p, by replacing every subpath τ_i and ρ_i with legal subpaths t_i and r_i. A formal description follows. For convenience, we use $s(\pi)$ and $f(\pi)$ do denote the start and the end of a path π (same for path p).

Proposition A.8. *Every relaxed path $\pi = \tau_0 \cdot \rho_1 \cdot \tau_1 \ldots \rho_k \cdot \tau_k$ with $s(\pi), f(\pi) \in X_v$ and starts and ends in X_v can be transformed into a legal path $p = t_0 \cdot r_1 \cdot t_1 \ldots r_k \cdot t_k$, where τ_i, ρ_i, t_i and r_i are defined as above and $s(\pi) = s(p)$ and $f(\pi) = f(s)$.*

Proof: The proof is by induction on k, the number of times the path π switches between nonempty tiles.

[**Basis, k = 0**]. If $\pi = \tau_0$, this implies that π starts and ends in the same conservative region and never leaves its original tile. From Corollary A.6 it follows that there exists a path $p = t_0$, such that $s(p) = s(\pi)$, $f(p) = f(\pi)$, and t_0 lies entirely within X_v. Thus p is a legal path.

[**Hypothesis, k < n**]. Assume that the proposition is true for all $k < n$.

[**Inductive step, k = n**]. Let $\pi = \tau_0 \cdot \rho_1 \cdot \tau_1 \ldots \rho_n \cdot \tau_n$ be as described in the proposition. We have to show that it can be transformed into a legal path $p = t_0 \cdot r_1 \cdot t_1 \ldots r_n \cdot t_n$. Let T_0 and T_1 be the first two nonempty tiles that π visits, i.e., $\tau_0 \subset T_0$ and $\tau_1 \subset T_1$. From Lemma A.7 it follows that there exists a pair of configurations $c_0 \in X_v \cap T_0$ and $c_1 \in X_v \cap T_1$ such that $r_0 = \overline{c_0c_1}$ is a horizontal segment in X, see Figure 8. Also, by the assumption, we have $s(\tau_0) = s(\pi) \in T_0 \cap X_v$, and since $c_0 \in T_0 \cap X_v$, from Corollary A.6 it follows that there exists a legal path $t_0 \subset T_0 \cap X_v$ with $s(t_0) = s(\tau_0)$ and $f(t_0) = c_0$. Finally, there is a path $\tau_1' \subset T_1$ such that $s(\tau_1') = c_1$ and $f(\tau_1') = f(\tau_1) = s(\rho_2)$.

We apply the inductive hypothesis to the relaxed path $\tau' = \tau_1' \cdot \rho_2 \cdot \tau_2 \ldots \rho_n \cdot \tau_n$ to get the legal path $p' = t_1 \cdot r_2 \cdot t_2 \ldots r_n \cdot t_n$. We define the legal path $p = t_0 \cdot r_0 \cdot p'$ thus proving the inductive step. ∎

Lemma A.9. *Every winning relaxed path can be transformed into a winning legal path.*

Proof: Follows immediately from Proposition A.8 and also from the fact that, for every nonempty tile T which borders the diagonal, the corresponding conservative region C also borders the diagonal. ∎