International Journal of Computational Geometry & Applications
© World Scientific Publishing Company

Clearing a Polygon with Two 1-searchers *

Borislav H. Simov
System Networking and Security Lab, Hewlett-Packard Co.
19410 Homestead Road, MS 4031, Cupertino, CA 95014, U.S.A.
boris_simov@hp.com

Giora Slutzki
Department of Computer Science, Iowa State University
208 Atanasoff Hall, Ames, IA 50011, U.S.A.
slutzki@cs.iastate.edu

Steven M. LaValle
Department of Computer Science, University of Illinois
201 N. Goodwin Ave., 3318 Siebel Center, Urbana, IL 61801, U.S.A.
lavalle@cs.uiuc.edu

Received 17 February 2003
Revised 14 November 2005
Communicated by (Name)

We present an algorithm for a pair of pursuers, each with one flashlight, searching for an unpredictable, moving target in a 2D environment (simple polygon). Given a polygon with n edges and m concave regions, the algorithm decides in time $O(n^2 + nm^2 + m^4)$ whether the polygon can be cleared by the two 1-searchers, and if so, constructs a search schedule. The pursuers are allowed to move on the boundary and in the interior of the polygon. They are not required to maintain mutual visibility throughout the pursuit.

Keywords: pursuit-evasion; visibility; motion planning; robotics; sensing

1. Introduction

Consider the following scenario: in a (dark, doorless) polygonal region there are three moving objects (represented as points). Two of them, called the **pursuers** (also known as **1-searchers**), have the task to find the third, called the **evader**. The evader can move arbitrarily fast, and his movements are unpredictable by the pursuers. Each pursuer is equipped with a flashlight and can see the evader only along the illuminated line segment emitted by the flashlight. The pursuers have perfect knowledge about each other's location. They plan their moves in cooperation

*An earlier version of this paper appeared at the 2002 IEEE International Conference on Robotics and Automation (ICRA'2002).

and are not required to maintain mutual visibility at all times. The pursuers **win** if they illuminate the evader with a flashlight. If there is a movement strategy of the pursuers whereby they win regardless of the strategy employed by the evader, we say that the polygon can be *cleared by two 1-searchers*.

The scenario above is a typical problem in pursuit-evasion, a field of continued interest in both robotics and computational geometry. The basic task in pursuit-evasion is to compute motion strategies for one or more pursuers to guarantee that unpredictable evaders will be detected. A key difficulty which makes the problem more challenging than basic exploration is that the evaders can sneak back to places already explored by the pursuers. Efficient algorithms that compute these strategies can be embedded in a variety of robotics systems to locate other robots and people. They can aid mobile surveillance systems that detect intruders using sonars, lasers, or cameras. Mobile robots can be used by special forces in high-risk military operations to systematically search a building in enemy territory before it is declared safe for entry.

In this paper we present an algorithm which, given a polygon with n edges and m concave regions, decides in time $O(n^2 + nm^2 + m^4)$ whether it can be cleared by the two 1-searchers, and if so, constructs a search schedule.

Note that for some polygons $m = \Theta(n)$, so the overall worst case running time of our algorithm is $O(n^4)$, if expressed in terms of n alone. Despite the fact that it is possible to design a somewhat simpler algorithm which runs in time $O(n^4)$, we have opted for an algorithm whose complexity depends explicitly on the number of concave regions m. The reason for this choice is that for most environments that arise in practice, the number of concave regions is much smaller than n and in these cases our algorithm performs much better than $O(n^4)$.

To make the motivation clearer, suppose that the actual environment is curved,[1] and a polygonal approximation is constructed. As the approximation quality is improved, the number, n, of edges increases; however, the number, m, of concave regions remains fixed. Thus, the running time of our algorithm depends more on the structure of the environment, as opposed to only counting edges. Therefore, in practice it is much faster than the naive $O(n^4)$ algorithm.

The rest of the paper is organized as follows. Section 1.1 gives a brief overview and history of the visibility-based pursuit-evasion problem. We introduce the notation and provide some basic definitions in Section 1.2. Two formal models of the pursuit as a search in a continuous information space are presented in Section 1.3 and Section 2. A discrete representation of the pursuit as a search in a finite graph is presented in Section 3. In Section 4 we present an algorithm which, given a polygon with n edges and m concave regions, decides in $O(n^2+nm^2+m^4)$ time whether the polygon can be cleared by two 1-searchers and if so, constructs a winning search schedule. We note that the representations in Sections 1.3 and 2 are given only as a conceptual framework and are not directly used by the algorithm in Section 4. Section 5 concludes the paper with a summary and directions for future research.

1.1. *Related work*

Pursuit-evasion in the plane was introduced by Suzuki and Yamashita.[2] They considered a single pursuer looking for an evader inside a simple polygon. They defined different kinds of pursuers depending on the number of flashlights that the pursuer is equipped with, e.g., a 1-searcher has one flashlight, a k-searcher has k flashlights, and an ∞-searcher has $360°$-vision. This naturally defines a pursuit-evasion problem for each class of searchers. Independently of Ref. 2, Icking and Klein defined the "two guard walkability problem",[3] which is a search problem for two guards whose starting and goal position are given, and who move on the boundary of a polygon while maintaining mutual visibility. A solution to the two guards problem was provided by Icking and Klein,[3] followed by improvements in Refs. 4, 5. While Refs. 1, 2, 6, 7, 8 presented polynomial solutions for deciding searchability of special classes of polygons, the general case single pursuer problem was open for quite a while.

Recently, the authors provided an $O(n^3)$ solution for a single 1-searcher in a polygon,[9] a result which they later improved to $O(n + m \log n + m^2)$ in Ref. 10. Park et al presented a polynomial solution for the case of a single 2-searcher and proved that adding more flashlights to a single pursuer does not increase the class of the polygons she can clear.[11] Note that the set of polygons that can be cleared by a single 2-searcher is a proper subset of the set of polygons that can be cleared by two 1-searchers. Figure 4(a) presents an example of a polygon which can be cleared by two 1-searchers, yet cannot be cleared by a single 2-searcher.

Guibas et al extended the pursuit-evasion problem to one in which multiple pursuers collaborate in order to clear a polygonal region.[12] They showed that determining the minimal number of pursuers needed to clear a polygonal region *with holes allowed* is an NP-hard problem. It is not known whether the same problem defined over simple polygons *without holes* is also NP-hard.

Efrat et al considered pursuit-evasion by a chain of k guards as a generalization of the search with two guards.[13] Their pursuit is subject to the restriction that the first and the k-th guards always move on the boundary while guard i, $1 < i < k$ moves in the interior of the polygon and maintains visibility with her neighbors, guards $i - 1$ and $i + 1$. Efrat et al gave a polynomial algorithm for the k guards problem.[13] Note that the pursuit with two 1-searchers is not a special case of the k guards pursuit since (i) the 1-searchers are not required to maintain visibility all the time, and (ii) for each 1-searcher, the endpoint of the ray of light emitted by her flashlight does not have to move continuously along the boundary of the polygon.

Suzuki et al provided a polynomial-time solution for a version of the pursuit in which a single ∞-searcher is restricted to the boundary of the polygonal region.[14] Interesting upper bounds on the number of searchers necessary and sufficient to clear a polygon were presented in Refs. 15, 16. Vidal et al addressed a version of the pursuit-evasion problem that uses probabilistic models.[17]

1.2. *Notation and preliminaries*

In the rest of the paper "pursuer" will be synonymous to a 1-searcher, unless otherwise specified. Also, all polygons are assumed to be simple. The **boundary** of a polygon P is denoted by ∂P and we assume that $\partial P \subseteq P$ and that ∂P is oriented in the **clockwise** (also called **positive**) direction. For two distinct points $a, c \in \partial P$, we write $\partial P(a, c)$ to denote the open interval of all points $b \in \partial P$ such that when starting after a in positive direction along ∂P, b is reached before c. We write $a \prec b \prec c$, if $b \in \partial P(a, c)$, and also use the notation $\partial P[a, c]$, $\partial P[a, c)$ and $\partial P(a, c]$ 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 positive direction. The **edges** of ∂P are e_0, e_1, …, e_{n-1}, where edge e_i has endpoints p_i and p_{i+1}, and the indices are computed modulo n, e.g., $p_n = p_0$. Vertex $p_i \in \partial P$ is a **reflex vertex** if the angle formed by incident edges e_{i-1} and e_i, in the interior of P, is greater than $180°$ (i.e., points p_{i-1}, p_i, and p_{i+1}, form a left turn). Otherwise, p_i is a **non-reflex vertex**.

We use a standard definition of visibility. For points $c, d \in P$ 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 no two points on the same edge of P are mutually visible.

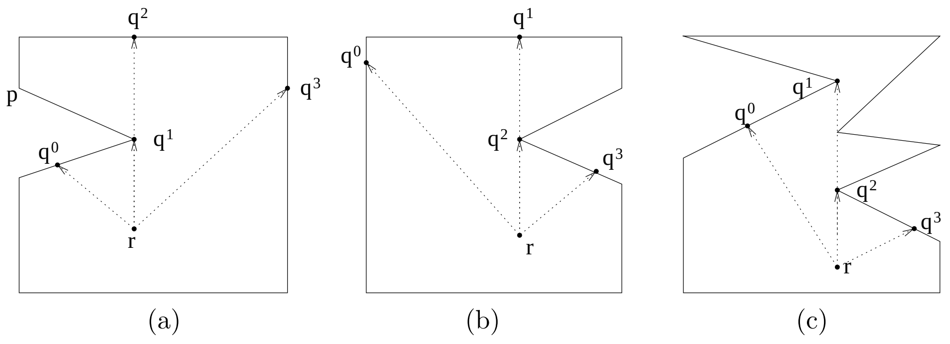


Fig. 1. The two kinds of gap edges: (a) left, (b) right. Part (c) illustrates a (right) gap edge in which q^1 and q^2 are not mutually visible.

Let $r \in P$, $q^1, q^2 \in \partial P$ be three colinear points, see Figure 1(a). We say that the pair (q^1, q^2) forms a **left gap edge** relative to r if:

- q^1 is visible from r,
- no point in $\partial P(q^1, q^2]$ is visible from r,
- every open interval which contains q^2 also contains a point visible from r.

Similarly, see Figure 1(b), the pair (q^2, q^1) forms a **right gap edge** relative to r if:

- q^2 is visible from r,
- no point in $\partial P[q^1, q^2)$ is visible from r,
- every open interval which contains q^1 also contains a point visible from r.

Note that as seen in Figure 1(c), the gap edge definitions do not require q^1 and q^2 to be mutually visible.

Clearly, the order of r, q^1 and q^2 is important in the definition of a gap edge. However, from now on we will be a bit more casual about the order, if it can be inferred from the context.

Consider $q^0, q^1, q^2, q^3 \in \partial P$ ordered in positive direction and $p \in P$. Suppose that r, q^1 and q^2 form a (left or right) gap edge and q^0 and q^3 are sufficiently close to q^1 and q^2 respectively, so that all the points in $(q^0, q^1) \cup (q^2, q^3)$ are visible from r, see Figure 1, (a) or (b). Suppose a pursuer is located at r and her **lightpoint** (the point of the boundary illuminated by her flashlight) is at q^0. If the pursuer rotates the flashlight clockwise, the lightpoint moves continuously over ∂P before it reaches q^1. At that moment the lightpoint jumps from q^1 to q^2. After q^2 the lightpoint moves continuously to q^3. We call this a **lightpoint jump** from (r, q^1) to (r, q^2). The reverse move, from (r, q^2) to (r, q^1), is also a lightpoint jump. We note that the lightpoint jumps are the only possible discontinuities in the location of the lightpoint.

1.3. *Semantics of the pursuit*

Within a polygon, the points which may contain the evader are defined to be **contaminated** and the rest of the points are defined to be **clear**.

For $i = 0, 1$ and mutually visible points $r_i \in P$, $q_i \in \partial P$, we say that pursuer i is in **configuration** $\langle r_i, q_i \rangle$ if the pursuer is located at r_i and her lightpoint is at q_i. Naturally then, the positions of both pursuers can be encoded as a pair of configurations, $\langle r_0, q_0 \rangle$ and $\langle r_1, q_1 \rangle$.

The segments $\overline{r_0 q_0}$ and $\overline{r_1 q_1}$ partition P into a number of connected components. We call each component a **contamination region**, consisting of all points of P which are connected by a path within P not crossed by a ray of light.[a] For example, in Figure 2(a) there are two contamination regions, C_0 and C_1. For stationary pursuers, an evader can move undetected to every point within a contamination region, hence all the points in that region have the same contamination status and we simply refer to the contamination region itself as being contaminated, or clear.

We can now define the motion of the pursuers as a trajectory in the space $P \times \partial P \times P \times \partial P$, parametrized over time. Without loss of generality we can assume that the pursuit starts at time $t = 0$ with the polygon contaminated. For $i = 0, 1$, define the functions $r_i : [0, \infty) \to P$, and $q_i : [0, \infty) \to \partial P$, such that at time t the i-th pursuer is in configuration $\langle r_i(t), q_i(t) \rangle$. For $i = 0, 1$, the functions

[a]To avoid tedious technicalities, we exclude from the partition the points lying on the segments $\overline{r_0 q_0}$ and $\overline{r_1 q_1}$.

r_i and q_i are subject to some additional constraints stemming from the semantics of the pursuit:

- At any time $t \in [0, \infty)$, $r_i(t)$ and $q_i(t)$ are mutually visible.
- The position of the pursuer, $r_i(t)$, is a continuous function.
- The lightpoint, $q_i(t)$, is a piecewise continuous function with discontinuities corresponding to the lightpoint jumps defined earlier.

We define a **schedule** to be a 4-tuple of functions, $(r_0(t), q_0(t), r_1(t), q_1(t))$, satisfying the above constraints. Starting with a contaminated polygon P, at any time $t \geq 0$ of the pursuit, a given schedule implicitly determines the contamination status of all the connected components of P, that is, at any time there is a well-defined set of points in P which are clear. If a schedule starts at time $t = 0$ with a contaminated polygon and at some time $t = T$, the polygon is clear, the schedule is called a **winning schedule**.

Definition 1. A polygon P can be cleared by two 1-searchers if there exists a winning schedule for P.

2. Canonical pursuit

In Section 1.3 we defined a model of the pursuit, called a schedule. Its main advantage is its simplicity, i.e., it explicitly represents the motions of the pursuers. However, it has certain shortcomings. First, a snapshot of the pursuit at a fixed moment of time t, or equivalently, the values $(r_0(t), q_0(t), r_1(t), q_1(t))$ alone, do not give us information about the complete status of the pursuit. We need the past trajectory of the four functions to determine the clear and contaminated areas of the polygon.

In this section we consider an equivalent model of pursuit, in which the two 1-searchers stay on the boundary *most of the time*. While the schedule defined in Section 1.3 corresponds to a trajectory in $(P \times \partial P)^2$, the schedule we define in Section 2.2 is similar to a trajectory in a smaller space, $(\partial P)^4$. Every snapshot of the new representation will encode the complete status of the pursuit: the pursuers' positions and the contaminated regions of the polygon.

2.1. *Canonical configurations*

We first show how, without causing contamination, we can map an arbitrary configuration to one in which the pursuer is on the boundary. For $i = 0, 1$, suppose pursuer i is at point $r_i \in P - \partial P$, directing the beam at point $q_i \in \partial P$, see Figure 2(b). Shoot a ray starting from r_i in the direction opposite of q_i and let $p_i \in \partial P$ be the first boundary point hit by the ray. We say that $\langle p_i, q_i \rangle$ is the **canonical configuration** corresponding to the configuration $\langle r_i, q_i \rangle$. If $r_i \in \partial P$, then $\langle r_i, q_i \rangle$ maps to itself.

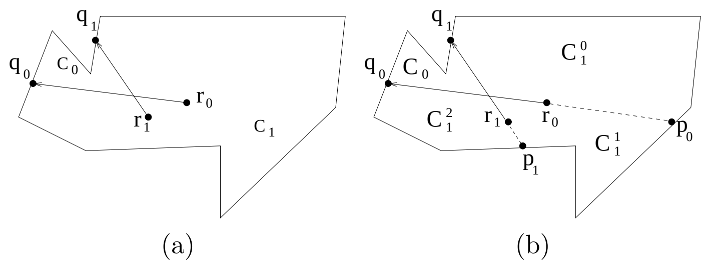


Fig. 2. Contamination regions induced: (a) by configurations $\langle r_0, q_0\rangle$ and $\langle r_1, q_1\rangle$ and (b) by the corresponding canonical configurations $\langle p_0, q_0\rangle$ and $\langle p_1, q_1\rangle$.

An interesting property of the mapping is that the contamination regions induced by the segments $\overline{p_iq_i}$ are a refinement of the regions induced by the segments $\overline{r_iq_i}$. For example, consider Figure 2(b) and assume the pursuers are in configurations $\langle r_i, q_i\rangle$, $i = 0, 1$. If we define the area $C_1 = C_1^0 \cup C_1^1 \cup C_1^2$, then $\overline{r_0q_0}$ and $\overline{r_1q_1}$ partition P into just two contamination regions, C_0 and C_1. On the other hand, for the corresponding canonical configurations, $\langle p_i, q_i\rangle$, $i = 0, 1$, there are four contamination regions: C_0, C_1^0, C_1^1 and C_1^2. The contamination regions C_0, C_1^0, C_1^1 and C_1^2 are a refinement of the regions C_0 and C_1 which leads us to the following remark.

Remark 1. At any single moment, we can replace a pair of pursuers located in the interior of P with the corresponding canonical pair without causing additional contamination.

We showed how to map an arbitrary configuration into a corresponding canonical one. A natural continuation would be to use the same transformation to map an arbitrary trajectory of a pursuer from $P \times \partial P$ onto $\partial P \times \partial P$. Most of the time the continuous motion of $r_i(t) \in P$ translates into a continuous motion of $p_i(t) \in \partial P$, see Figure 3(a). However, there are exceptions: p_i may not be a continuous function on ∂P, as seen in the example in Figure 3(b). For simplicity, assume that pursuer 0 is stationary at point r_0 with lightpoint q_0. Pursuer 1 moves over a continuous path from r_1^1 to r_1^4, which is projected as a **piecewise continuous** path on ∂P from p_1^1 to p_1^4. The move of the first pursuer can be divided into two parts. The first part, from r_1^1 to r_1^2, is projected into the continuous path from p_1^1 to p_1^2. The second part, from and excluding r_1^2 to r_1^4, is projected into the continuous path from and excluding p_1^3 to p_1^4. Note that the jump from p_1^2 to p_1^3 represents a discontinuity in $p_1(t)$, the projection of $r_1(t)$ on ∂P, and this jump cannot be simulated by a pursuer moving solely over the boundary.

The solution is to allow the pursuer to move along the segment $\overline{p_1^2p_1^3}$. Thus the continuous motion from r_1^1 to r_1^4 can be represented as a continuous motion from p_1^1 to p_1^2 along ∂P, followed by a continuous motion from p_1^2 to p_1^3 inside P, followed by a continuous motion from p_1^3 to p_1^4 along ∂P. Note that technically, we do not want the motion from p_1^2 to p_1^3 to be exactly along the segment $\overline{p_1^2p_1^3}$ since q^1 will not be

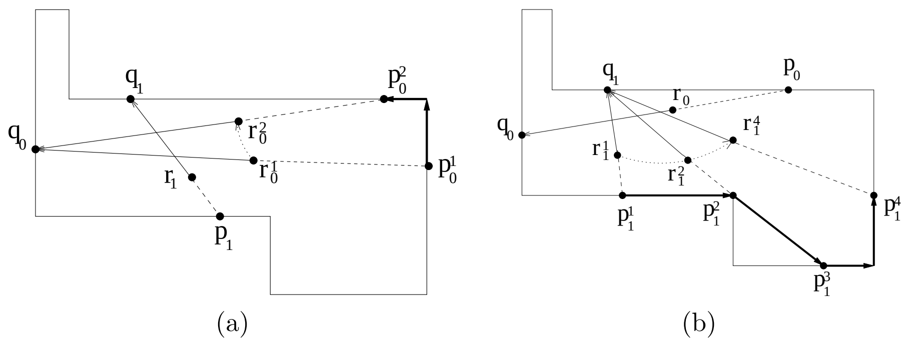


Fig. 3. Projecting motion in the interior onto motion on the boundary.

visible along the segment. Instead, the move is performed along a segment $\overline{p_1^2 p^*}$, where $p^* \in \partial P$ is sufficiently close to p_1^3 so that every point from $\overline{p_1^2 p^*}$ is visible from q_1.

We call the move from $\langle p_1^2, q_1 \rangle$ to $\langle p_1^3, q_1 \rangle$ a **pursuer jump**. Just like in the case of the lightpoint jump, the reverse move, from $\langle p_1^3, q_1 \rangle$ to $\langle p_1^2, q_1 \rangle$, is also considered a pursuer jump. Note that in reality the physical location of the pursuer is still continuous but at least for a moment the pursuer had left the boundary. From now on this will be the only circumstance in which the pursuers will leave ∂P.

2.2. *Canonical schedule*

Our next goal is to define a pursuer schedule based on canonical configurations. Recall that we defined canonical configurations as *directed* line segments of mutually visible points on the boundary. However a lot of the discussion in the rest of the paper will deal with visibility, which is a symmetric relation. Therefore, we do not base the new schedules solely on the original definition of canonical configurations, but we also incorporate an alternative interpretation of canonical configurations as *undirected* line segments.

Let x_0, y_0, x_1, $y_1 \in \partial P$, such that $x_0 \prec x_1 \prec y_1$, and for $i \in \{0, 1\}$, x_i and y_i are mutually visible. Let the **order** (of y_0), k, denote the number of points from $\{x_1, y_1\}$ between x_0 and y_0, or:

$$k = |\{x_1, y_1\} \cap \partial P(x_0, y_0)| = \begin{cases} 0, & \text{if } x_0 \prec y_0 \prec x_1 \prec y_1 \\ 1, & \text{if } x_0 \prec x_1 \prec y_0 \prec y_1 \\ 2, & \text{if } x_0 \prec x_1 \prec y_1 \prec y_0 \end{cases}$$

We refer to (x_0, y_0, x_1, y_1, k) as a **visibility tuple**. Due to the circularity of ∂P, each tuple has three more equivalent tuples: (y_0, x_1, y_1, x_0, k'), $(x_1, y_1, x_0, y_0, k'')$, and $(y_1, x_0, y_0, x_1, k''')$. The set of all visibility tuples is defined as the **visibility space**, $\mathcal{I}_v$.

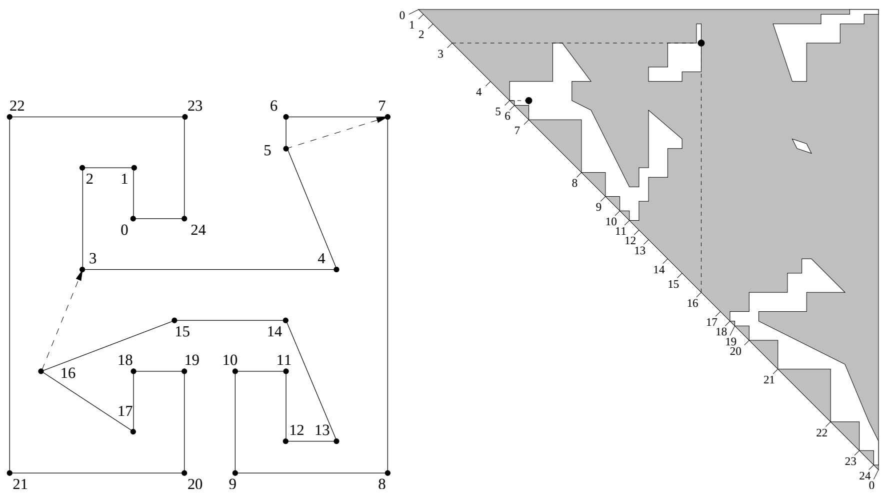


Fig. 4. Visibility obstruction diagram (b) for the polygon in (a). The white regions in (b) correspond to the set $\mathcal{X}_v$ of the mutually visible pairs of points in ∂P. The two pursuers are located at points 16 and 5 of the polygon in (a) and illuminate points 3 and 7 respectively. The positions of the pursuers are shown as the black circles at points (5, 7) and (3, 16) in the visibility obstruction diagram (b).

Consider a visibility tuple (x_0, y_0, x_1, y_1, k) and suppose $\langle p_0, q_0\rangle$ and $\langle p_1, q_1\rangle$ are two configurations with the property the light segments $\overline{p_0q_0}$ and $\overline{p_1q_1}$ are equivalent to the light segments $\overline{x_0y_0}$ and $\overline{x_1y_1}$ up to reordering of the endpoints of the segments and swapping the indices. The visibility tuple contains exactly the same *visibility* information as the pair of configurations $\langle p_0, q_0\rangle$, $\langle p_1, q_1\rangle$. At the same time, the visibility tuple has the advantage that the order of the points x_0, y_0, x_1, and y_1 is completely determined by their position in the tuple and k.

Let 0 be an arbitrary point from the boundary. Denote with $\mathcal{X}$ the set of all pairs of points $(x, y) \in \partial P \times \partial P$, such that $0 \preceq x \preceq y \prec 0$. Denote with $\mathcal{X}_v$ the set of all pairs $(x, y) \in \mathcal{X}$, such that x and y are mutually visible. The set $\mathcal{X}_v$ is part of the visibility obstruction diagram (VOD) defined in Ref. 9, as a graphical representation of the visibility relation between pairs of points from ∂P. The only difference is that, since visibility is a symmetric relation, without loss of generality, in this paper we have restricted $\mathcal{X}_v$ to the points above the diagonal. For example, consider the polygon in Figure 4(a). The corresponding VOD $\mathcal{X}_v$ is shown as the set of points in the white regions in Figure 4(b). If for the moment we disregard k, then we can think about a visibility tuple merely as two points in $\mathcal{X}_v$. A visibility tuple $(3, 16, 5, 7, 2)$ denotes that the two light segments are $\overline{3, 16}$ and $\overline{5, 7}$ but does not determine the position of the pursuers, i.e., pursuer 0 can be either at point 3 or at point 16 of the polygon in Figure 4(a). Points $(3, 16)$ and $(5, 7)$ from the set $\mathcal{X}_v$ shown in Figure 4(b) represent the positions of the two light segments $\overline{3, 16}$ and

$\overline{5,7}$.

We have replaced a pair of configurations $\langle p_0, q_0 \rangle$ and $\langle p_1, q_1 \rangle$, i.e., two directed segments, with a visibility tuple (x_0, y_0, x_1, y_1, k), which no longer stores the direction of the segments, or equivalently, the location of the pursuer. To record this additional information, we need one extra bit for each segment. For $i \in \{0, 1\}$, define the **direction bit** d_i for pursuer i, such that $d_i \in \{0, 1\}$, and $d_i = 0$, when x_i corresponds to a pursuer location, i.e., when $x_i \in \{p_0, p_1\}$.

As we mentioned before, one drawback of the schedule defined in Section 1.3 is that it does not explicitly show the status of the contamination regions. We fix this as follows. Since the segments $\overline{p_i q_i}$, or equivalently, $\overline{x_i y_i}$, $i = 0, 1$, divide the boundary into four regions, four bits will be sufficient to record the contamination status at a given moment of time. Let us number the intervals on the boundary from 0 to 3 in positive direction, starting from the interval beginning at x_0. For $i = 0 \ldots 3$, define **contamination bit** $b_i \in \{0, 1\}$, to be the contamination status of interval i, with $b_i = 0$, when the interval is clear. We refer to $(d_0, d_1, b_0, b_1, b_2, b_3)$ as a **bits tuple**. The set of all bits tuples is defined as the **bits space**, $\mathcal{I}_b$.

Consider a fixed moment during a pursuit, and suppose that the two pursuers are in configurations $\langle p_i, q_i \rangle$, $i = 0, 1$. Let k, x_i, y_i, d_i, $i = 0, 1$, and b_j, $j = 0 \ldots 3$ be as defined above. Define the **canonical information state** for the pursuit to be the concatenation of the visibility and the bits tuples:

$$(x_0, y_0, x_1, y_1, k, d_0, d_1, b_0, b_1, b_2, b_3)$$

Note that the information state gives us explicitly a full snapshot of the pursuit at that moment. Consider again the polygon in Figure 4(a) and its corresponding VOD, Figure 4(b). A canonical information state $(3, 16, 5, 7, 2, 1, 0, 0, 1, 0, 1)$ denotes that the two pursuers are at points 16 and 5, illuminating points 3 and 7 respectively. The intervals $\partial P(5, 7)$ and $\partial P(16, 3)$ are contaminated. The intervals $\partial P(3, 5)$ and $\partial P(7, 16)$ correspond to the same region which is clear.

A canonical information state is a **start** if all the contamination bits are 1, corresponding to a contaminated polygon. A canonical information state is a **goal** if the contamination bits are 0, i.e., the polygon is clear. The set of all canonical information states is the **canonical information space**, $\mathcal{I}$. Observe that $\mathcal{I} = \mathcal{I}_v \times \mathcal{I}_b$, i.e., $\mathcal{I}$ is a product of the infinite space $\mathcal{I}_v$ and the finite space $\mathcal{I}_b$.

After the definition of information states and the introduction of the pursuer jump we can define a **canonical schedule** to be a piecewise continuous trajectory in the corresponding canonical information space, $\mathcal{I}$, parametrized over time. The canonical schedule is quite similar to a schedule with the additional restriction that the pursuers most of the time move on the boundary and enter the interior only during the pursuer jumps. We can view a canonical schedule as a function $I : [0, \infty) \to \mathcal{I}$, where $I(t) = (x_0, y_0, x_1, y_1, k, d_0, d_1, b_0, b_1, b_2, b_3)$ encodes the information state at time t. Just like in the previous definitions, a canonical schedule has to also satisfy some conditions derived from the semantics of the pursuit:

- The functions $x_i(t)$, $y_i(t)$ are piecewise continuous in ∂P, with discontinuities corresponding to pursuer or lightpoint jumps. The direction bit d_i is required to determine the type of a jump for $x_i(t)$ or $y_i(t)$.
- At time 0 the canonical schedule is in a starting canonical information state.
- The direction bits d_i, the contamination bits b_j and the order k change only during a jump or during a change in the relative order of x_0, y_0, x_1, y_1. (We provide a detailed explanation of the types of changes to the information states in Section 2.3.)

A canonical schedule which at some time reaches a goal state is called a **winning canonical schedule**.

In the rest of this section we show that winning schedules are equivalent to the original schedules. That is, we do not reduce the power of the pursuers by restricting them to moving on the boundary most of the time.

Lemma 1. *A polygon can be cleared by two 1-searchers, if and only if there exists a winning canonical schedule.*

Proof. First, consider the reverse direction. Note that if we ignore the details of the representation, every canonical schedule is just a more restricted version of a schedule. So if there exists a winning canonical schedule, then there also exists a winning schedule, therefore the polygon can be cleared by two 1-searchers.

For the forward direction, assume that the polygon can be cleared, so there exists a winning schedule. Consider its corresponding canonical schedule. From Remark 1, at any time, the area of the polygon which is cleared by the canonical schedule is equal to or includes the area cleared by the original schedule. If the original schedule is a winning one, then at some point it will clear the entire polygon. It follows that the area cleared by the corresponding canonical schedule is the whole polygon as well, therefore the canonical schedule is also a winning one. □

2.3. *Changes to the canonical information states*

In this section we discuss the ways contamination bits change. This will allow us to define a finite set of elementary moves, so that a canonical schedule can be considered a sequence of these elementary moves.

From the definition of a canonical schedule, it follows that it is a piecewise continuous function in $\mathcal{I}$. If we consider every element of the tuple $I(t)$ as a function of time, for most of the duration of the pursuit, it is a continuous function. Only at discrete moments of time, there are relative order changes or discontinuities in x_0, y_0, x_1, or y_1. These conditions trigger corresponding changes in the other elements of the tuple, the direction bits d_i, the contamination bits b_j and the order k. We identify the different types of changes to the information state and define those as **elementary moves**. For a given schedule, let $0 < t_1 < t_2 < \ldots$ be the points of time at which either a jump or a change of order occurs. We call each portion (t_{i-1}, t_i) of the schedule a type 1 move. Every type 1 move corresponds to a continuous path in the canonical information space, $\mathcal{I}$. On the other hand, the change occurring at each t_i corresponds to a jump in $\mathcal{I}$. Details about the different types of elementary moves are provided in the rest of the section.

We assume that at the beginning of the move, the information state is

$$I' = (x'_0, y'_0, x'_1, y'_1, k', d'_0, d'_1, b'_0, b'_1, b'_2, b'_3) \ ,$$

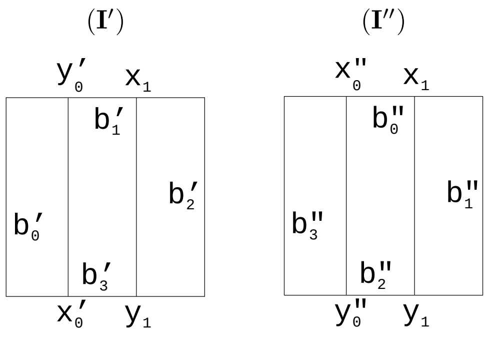
and at the end of the move the information state is

$$I'' = (x''_0, y''_0, x''_1, y''_1, k'', d''_0, d''_1, b''_0, b''_1, b''_2, b''_3) \ .$$

In the following paragraphs we denote the move from I' to I'' as $(I' \Rightarrow I'')$ and we describe the move by defining the value of I'' as a function of I'. Also, we note that the reverse move, from I'' to I', is also an elementary move. We denote it as $(I' \Leftarrow I'')$ and we describe it by defining the value of I' as a function of I''. For simplicity, if any of the x_i's or the y_i's are the same in I' and in I'', we omit the $'$ symbol. An exhaustive list of all the elementary moves follows:

Type 0: change of choice of x_0

This elementary move is purely technical. It represents no real change in the canonical information state, merely a relabeling of the positions of the pursuers and the bits. Recall that in the definition of a canonical information state in Section 2.2 we chose x_0 arbitrarily out of $\{p_0, q_0, p_1, q_1\}$. This implies that depending on the choice of x_0, there are four different canonical information states which represent the same status of the pursuit. In order to account for that fact, we define a move which switches x_0 to be the next point in positive direction out of x_1, y_0. (Note that we do not have to consider y_1 as a next point since it is always after x_1 in positive direction.) The following figure is an example of a type 0 move from a canonical configuration with $k = 0$.

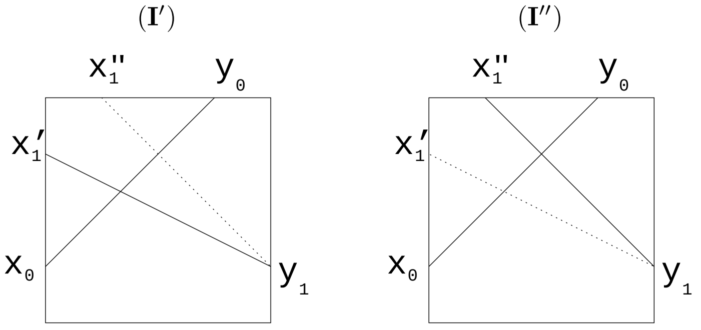


Let I' be the information state before the switch. Depending on the value of k', after the switch the value of I'' is as follows:

$$(I' \Rightarrow I''):\quad I'' = \begin{cases} (y'_0, x'_0, x'_1, y'_1, 1-d'_0,\ \ d'_1,\ \ 2, b'_1, b'_2, b'_3, b'_0), & \text{if } k'=0 \\ (x'_1, y'_1, y'_0, x'_0,\ \ d'_1,\ \ 1-d'_0, 1, b'_1, b'_2, b'_3, b'_0), & \text{if } k'=1 \\ (x'_1, y'_1, y'_0, x'_0,\ \ d'_1,\ \ 1-d'_0, 0, b'_1, b'_2, b'_3, b'_0), & \text{if } k'=2 \end{cases}$$

Type 1: no jumps, no change in the relative order

This is a continuous move which occupies most of the time of the canonical schedule. Throughout the duration of the move there are no jumps. Both the relative order of x_0, y_0, x_1, y_1 and the values of the bits are preserved. The next figure provides an example for a type 1 move with $k = 1$. The move is similar for the other values of k.



Apart from the location of the points x_0, y_0, x_1, y_1, the other parameters of the information states do not change:

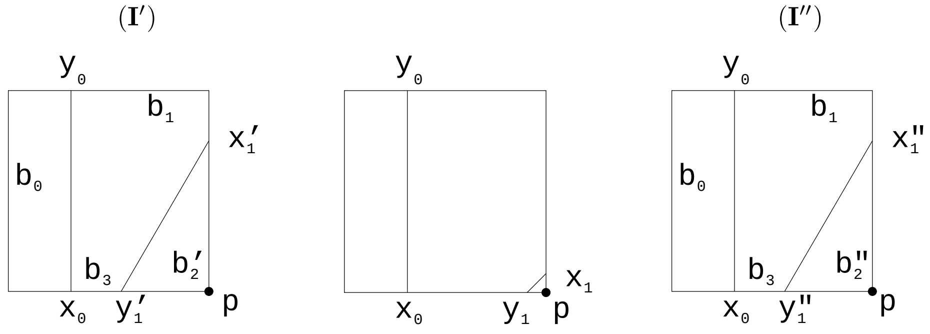
$$\begin{array}{ll} (I' \Rightarrow I''): & k'' \leftarrow k', b''_j \leftarrow b'_j, j \in Z_3 \\ (I' \Leftarrow I''): & k' \leftarrow k'', b'_j \leftarrow b''_j, j \in Z_3 \end{array}$$

A type 1 move followed immediately by its own reverse, always returns to the original information state.

Type 2: clearing a corner

The move occurs at a moment in which one of the light segments, $\overline{x_i y_i}$, becomes arbitrarily small, i.e., during the move points x_i and y_i merge at (or more precisely, converge arbitrarily close on both sides of) a non-reflex vertex, p.

Since the type 0 move allows us to relabel the points, without loss of generality, we can assume that x_1 and y_1 merge at vertex $p \in \partial P(x_1, y_1)$, as shown in the example.

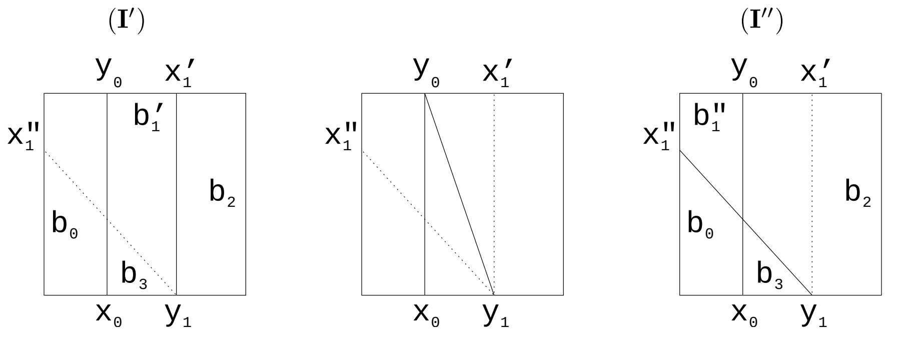


Since the underlying contamination region can be made arbitrarily small, we assume that after the move the region incident to p is clear. The forward and the reverse move are identical, so we only describe one:

$$(I' \Rightarrow I''): \quad k'' = k',\ b''_2 \leftarrow 0,\ b''_j \leftarrow b'_j, j \in \{0,1,3\}$$

Type 3: point merge move

The move occurs at a moment in which two endpoints of different light segments merge into a single one and subsequently their relative order changes. As mentioned before, the type 0 move allows us without loss of generality to choose x_0 such that $k' = 0$ and the merge is between y_0 and x_1. In the forward direction, the move represents a change in the relative order from $x_0 \prec y_0 \prec x'_1 \prec y_1$ to $x_0 \prec x''_1 \prec y_0 \prec y_1$, i.e., y_0 and x_1 switch positions.



At the moment in which $y_0 = x_1$ there is a change in the contamination regions. There were three regions prior to that moment and four regions after that. The newly created region is clear. The tuples change as follows:

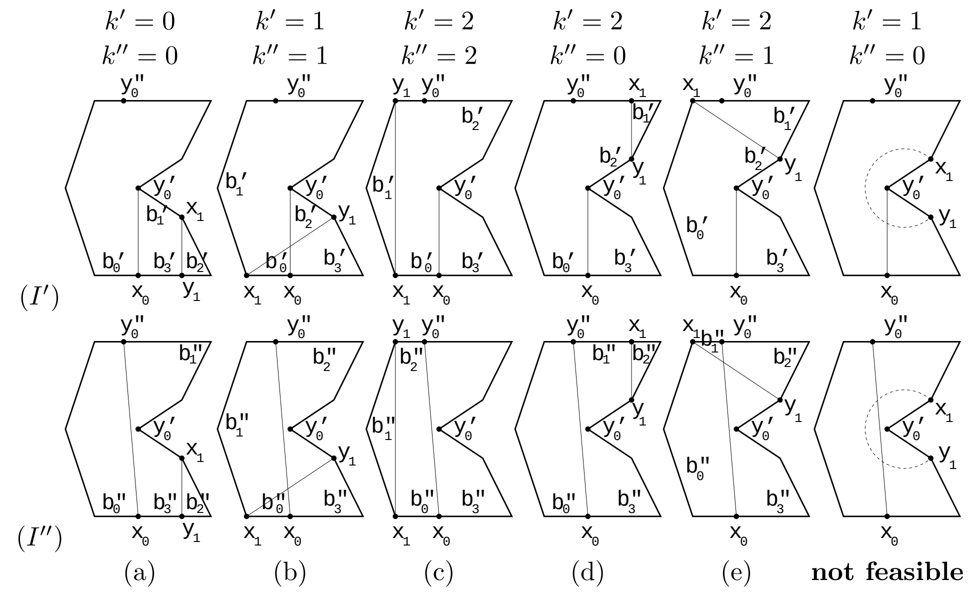
$$(I' \Rightarrow I''): k'' = 1,\ b''_1 \leftarrow 0,\ b''_j \leftarrow b'_j, j \in \{0,2,3\}$$
$$(I' \Leftarrow I''): k' = 0,\ b'_1 \leftarrow b''_3,\ b'_j \leftarrow b''_j, j \in \{0,2,3\}$$

Type 4 and Type 5: lightpoint and pursuer jumps

Next we describe moves of type 4 and type 5. They are both similar in the sense that one of the pursuers makes a pursuer or lightpoint jump, while the other pursuer is stationary. The jump results in possible contamination of previously clear regions. (We call this **recontamination**.) There may also be a simultaneous change in the relative order of the points x_0, y_0, x_1, and x_1.

Without loss of generality, we can choose x_0 such that (i) pursuer 1 is stationary (ii) y_0' is an interior point of the line segment $\overline{x_0 y_0''}$, (iii) there is a jump from y_0' to y_0''. The jump is a lightpoint one if and only if $d_0 = 0$. Otherwise, if $d_0 = 1$, the jump is a pursuer one.

The way a jump will affect the values of k and b_j depends on the relative position of x_0, y_0', y_0'', x_1 and y_1, i.e, on the old and new values of k: $k', k'' \in \{0, 1, 2\}$. Without loss of generality we can assume that $k' \geq k''$ which leaves 6 different combinations for k' and k''. One of the combinations ($k' = 1$, $k'' = 0$) is infeasible, the other five are shown as follows:



Type 4: lightpoint jump

In the forward direction of this move, ($I' \Rightarrow I''$), there is a single lightpoint jump, from y_0' to y_0'', i.e., $d_0 = 0$. The changes to the tuples are listed for each of the 5 possible cases:

$$\begin{array}{rll}
(a)\ (I' \Rightarrow I''): & k'' \leftarrow 0, b''_3 \leftarrow b''_1 \leftarrow b'_1 \vee b'_0, b''_j \leftarrow b'_j, j \in \{0,2\} \\
(I' \Leftarrow I''): & k' \ \leftarrow 0, b'_0 \leftarrow b''_0 \vee b''_1, b'_j \leftarrow b''_j, j \in \{1,2,3\} \\
(b)\ (I' \Rightarrow I''): & k'' \leftarrow 1, b''_2 \leftarrow b'_2 \vee b'_1, b''_j \leftarrow b'_j, j \in \{0,1,3\} \\
(I' \Leftarrow I''): & k' \ \leftarrow 1, b'_1 \leftarrow b''_1 \vee b''_2, b'_j \leftarrow b''_j, j \in \{0,2,3\} \\
(c)\ (I' \Rightarrow I''): & k'' \leftarrow 2, b''_3 \leftarrow b'_3 \vee b'_2, b''_j \leftarrow b'_j, j \in \{0,1,2\} \\
(I' \Leftarrow I''): & k' \ \leftarrow 2, b'_0 \leftarrow b'_2 \leftarrow b''_2 \vee b''_3, b'_j \leftarrow b''_j, j \in \{1,3\} \\
(d)\ (I' \Rightarrow I''): & k'' \leftarrow 0, b''_1 \leftarrow b''_3 \leftarrow b'_2 \vee b'_3, b''_0 \leftarrow b'_0, b''_2 \leftarrow b'_1 \\
(I' \Leftarrow I''): & k' \ \leftarrow 2, b'_2 \leftarrow b'_0 \leftarrow b''_0 \vee b''_1, b'_3 \leftarrow b''_3, b'_1 \leftarrow b''_2 \\
(e)\ (I' \Rightarrow I''): & k'' \leftarrow 1, b''_2 \leftarrow b''_1 \leftarrow b'_1, b''_3 \leftarrow b'_3 \vee b'_2, b''_0 \leftarrow b'_0 \\
(I' \Leftarrow I''): & k' \ \leftarrow 2, b'_2 \leftarrow b'_0 \leftarrow b''_0 \vee b''_3, b'_1 \leftarrow b''_1 \vee b''_2, b'_3 \leftarrow b''_3
\end{array}$$

Type 5: pursuer jump

In the forward direction of this move, $(I' \Rightarrow I'')$, there is a single pursuer jump, from y'_0 to y''_0, i.e., $d_0 = 1$. The changes to the tuples are listed for each of the 5 possible cases:

$$\begin{array}{rll}
(a)\ (I' \Rightarrow I''): & k'' \leftarrow 0, b''_0 \leftarrow b''_1 \leftarrow b''_3 \leftarrow b'_0 \vee b'_1, b''_2 \leftarrow b'_2 \\
(I' \Leftarrow I''): & k' \ \leftarrow 0, b'_0 \leftarrow b'_1 \leftarrow b'_3 \leftarrow b''_0 \vee b''_1, b'_2 \leftarrow b''_2 \\
(b)\ (I' \Rightarrow I''): & k'' \leftarrow 1, b''_1 \leftarrow b''_2 \leftarrow b'_1 \vee b'_2, b''_j \leftarrow b'_j, j \in \{0,3\} \\
(I' \Leftarrow I''): & k' \ \leftarrow 1, b'_1 \leftarrow b'_2 \leftarrow b''_1 \vee b''_2, b'_j \leftarrow b''_j, j \in \{0,3\} \\
(c)\ (I' \Rightarrow I''): & k'' \leftarrow 2, b''_0 \leftarrow b''_2 \leftarrow b''_3 \leftarrow b'_2 \vee b'_3, b''_1 \leftarrow b'_1 \\
(I' \Leftarrow I''): & k' \ \leftarrow 2, b'_0 \leftarrow b'_2 \leftarrow b'_3 \leftarrow b''_2 \vee b''_3, b'_1 \leftarrow b''_1 \\
(d)\ (I' \Rightarrow I''): & k'' \leftarrow 0, b''_0 \leftarrow b''_1 \leftarrow b''_3 \leftarrow b'_3 \vee b'_2, b''_2 \leftarrow b'_1 \\
(I' \Leftarrow I''): & k' \ \leftarrow 2, b'_0 \leftarrow b'_2 \leftarrow b'_3 \leftarrow b''_0 \vee b''_1, b'_1 \leftarrow b''_2 \\
(e)\ (I' \Rightarrow I''): & k'' \leftarrow 1, b''_2 \leftarrow b''_1 \leftarrow b'_1, b''_0 \leftarrow b''_3 \leftarrow b'_0 \vee b'_3 \\
(I' \Leftarrow I''): & k' \ \leftarrow 2, b'_1 \leftarrow b''_1 \vee b''_2, b'_0 \leftarrow b'_2 \leftarrow b'_3 \leftarrow b''_0 \vee b''_3
\end{array}$$

3. Finite representation

In the previous section we defined canonical configurations as a simpler model for the two pursuer problem. Yet the canonical information space is still an infinite one, which makes an exhaustive search for a winning canonical schedule infeasible. In this section we will introduce an equivalent, finite representation of the search space and we will show how to find a winning strategy by an exhaustive search in the finite space.

3.1. *VHC boxes*

Let $X = \partial P(\underline{x}, \overline{x})$ and $Y = \partial P(\underline{y}, \overline{y})$ be two open intervals from ∂P such that $\underline{x} \prec \overline{x} \prec \overline{y}$ and $\underline{x} \prec \underline{y} \prec \overline{y}$. Let $B(X, Y)$ be the region from $\mathcal{X}$ defined as $\{(x, y) \mid x \in X, y \in Y, \underline{x} \prec x \prec y\}$ and define $C(X, Y) = B(X, Y) \cap \mathcal{X}_v$. We say that $B(X, Y)$ is a **vertically and horizontally convex (VHC) box**, and that $C(X, Y)$ is a **VHC core**, if the following conditions hold:

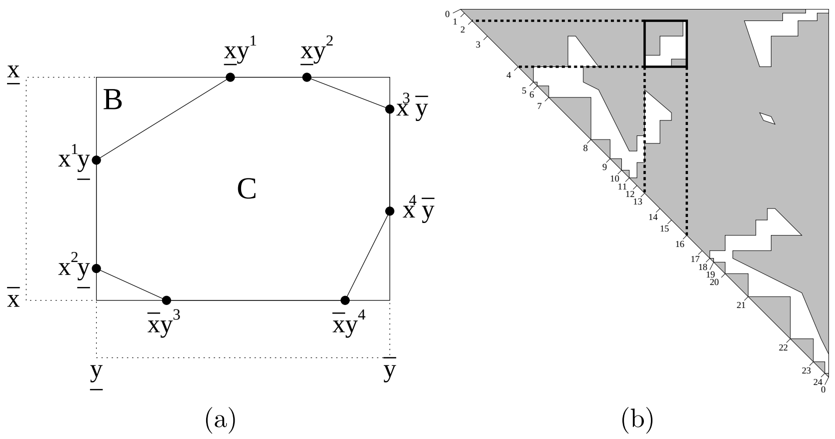


Fig. 5. Part (a) is an example of a VHC box B and core C with the extreme points denoted as black circles. Part (b) shows the VOD of the polygon from Figure 4(a). The intervals $X = (2, 4)$ and $Y = (13, 16)$ define a VHC box which is shown as a rectangle with thick solid lines. The corresponding VHC core is the white area inside the VHC box.

- $C(X, Y)$ is non-empty.
- for every $x^* \in X$, the set $\{y \mid (x^*, y) \in C(X, Y)\}$ is a single connected nonempty interval. This condition corresponds to **horizontal convexity**.
- for every $y^* \in Y$, the set $\{x \mid (x, y^*) \in C(X, Y)\}$ is a single connected nonempty interval. This condition corresponds to **vertical convexity**.

We define eight parameters or **extreme points**, which will be sufficient to represent $C(X, Y)$. Let $\underline{x}$ be the lower boundary of the interval X. If $(\underline{x}, y^1)$ and $(\underline{x}, y^2)$ are the endpoints of the segment $\{(\underline{x}, y) \mid (\underline{x}, y) \in C(X, Y)\}$, we call them the extreme points for $\underline{x}$. Similarly, there are three other pairs of extreme points, two points for each $\overline{x}$, $\underline{y}$ and $\overline{y}$. For example of a VHC box, VHC core, and corresponding parameters, refer to Figure 5. In the rest of the paper, we will not be interested in the precise shape of $C(X, Y)$. Instead, the points $\underline{x}$, $\overline{x}$, $\underline{y}$, $\overline{y}$, x^1, x^2, x^3, x^4, y^1, y^2, y^3, y^4 will be sufficient to record the extremums of $C(X, Y)$. Note that the extreme points need not be distinct and it is possible for a box to have as few as two unique extreme points.

For $i = 0, 1$, let $X_i = (\underline{x_i}, \overline{x_i})$, and $Y_i = (\underline{y_i}, \overline{y_i})$ be open intervals on the boundary ∂P, defining the VHC box $B(X_i, Y_i)$. Consider the intersections $X_0 \cap X_1$, $X_0 \cap Y_1$, $Y_0 \cap X_1$ and $Y_0 \cap Y_1$. If at most one of the four intersections is nonempty, we say that the two boxes are **independent**. Otherwise, we say that the boxes are **dependent**. See Figure 6 for examples of the two kinds. In part (a), consider $X_0 = (a, c)$, $Y_0 = (b, d)$, $X_1 = (e, f)$ and $Y_1 = (g, h)$, defining boxes B_0 and B_1 correspondingly. Neither X_0 nor Y_0 intersect with X_1 or Y_1, therefore the boxes B_0 and B_1 are independent. In part (b) of the same figure, consider $X_0 = (a, b)$, $Y_0 = (e, f)$, $X_1 = (c, g)$ and $Y_1 = (d, h)$, defining boxes B_0 and B_1 correspondingly.

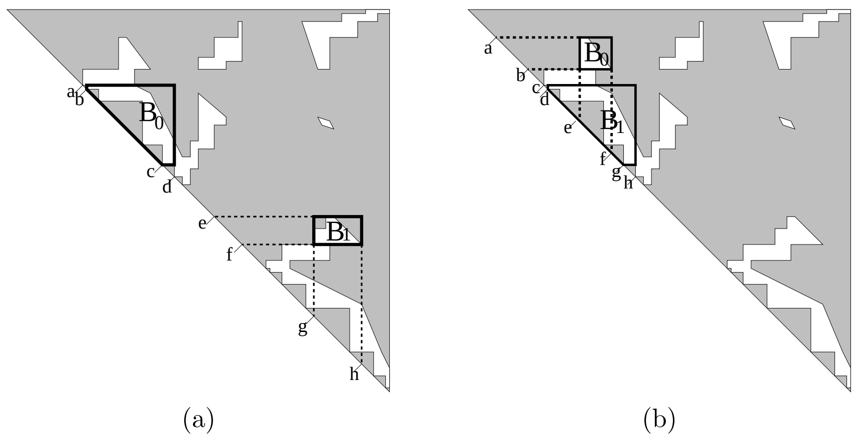


Fig. 6. Part (a) is an example of independent VHC boxes. Part (b) is an example of dependent boxes.

Since Y_0 intersects with both X_1 and Y_1, it follows that B_0 and B_1 are dependent boxes.

The next lemma allows us to determine in constant time whether a particular visibility tuple is feasible.

Lemma 2. *Let X_0, Y_0, X_1, and Y_1 be open intervals on the boundary ∂P, defining VHC boxes $B(X_0, Y_0)$ and $B(X_1, Y_1)$. Given an integer $k \in \{0, 1, 2\}$ and the parameters of the two boxes, define the condition*

$$\exists(x_i, y_i) \in C(X_i, Y_i), 0 \leq i \leq 1 : I_v = (x_0, y_0, x_1, y_1, k) \in \mathcal{I}_v \quad . \tag{1}$$

The condition above can be evaluated in $O(1)$ time. If existing, a tuple I_v can be constructed in time $O(1)$ if the boxes are independent and in time $O(n)$ if the boxes are dependent.

Proof. The proof is presented in the appendix. □

3.2. *Concave regions and critical intervals*

Let p_i and p_j be two different non-reflex vertices from ∂P which are not adjacent. If all the vertices $p_{i+1}, \ldots, p_{j-1}$ are reflex vertices, we say that p_i and p_j are **critical points** and we refer to the subinterval of ∂P of the form $\partial P(p_i, p_j)$ as a **concave region**. [b] We denote by m the number of concave regions in the polygon. So in a polygon P there are $2m$ critical points which divide ∂P into $2m$ intervals called **critical intervals**. Obviously, two concave regions cannot overlap and must be

[b]The terms concave regions and critical points of ∂P have been defined previously, see Ref. 18 for more details.

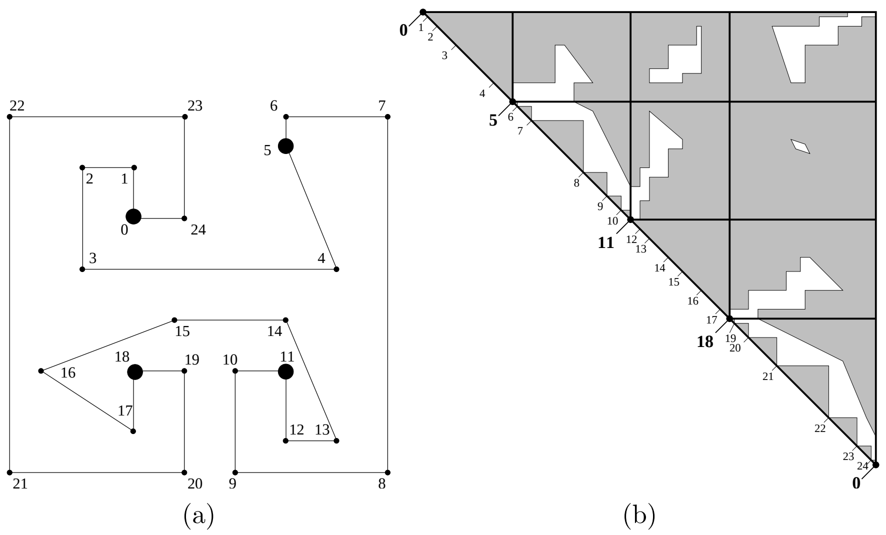


Fig. 7. The polygon in part (a) has critical points 0, 5, 11 and 18 and concave regions $\partial P(0,5)$ and $\partial P(11,18)$. Part (b) shows the VOD of the same polygon with the grid created by the critical points.

separated by non-concave regions (each of which contains at least one non-reflex vertex). For example, for the polygon in Figure 7(a) we have $m = 2$, the critical points are 0, 5, 11 and 18 and the concave regions are $\partial P(0,5)$ and $\partial P(11,18)$.

The critical points define a grid of vertical and horizontal lines over $\mathcal{X}$ which partition the set of mutually visible points $\mathcal{X}_v$ into multiple maximal connected subsets. The next lemma establishes a connection between the elements of this grid and the VHC cores defined in Section 3.1.

Lemma 3. *Consider the grid over $\mathcal{X}$ defined by the critical points and let A be an arbitrary grid element. If C is the intersection of the interior of $\mathcal{X}_v$ with the interior of A, then C is either empty or it is a VHC core.*

Proof. The proof follows directly from Lemma 3.9 of Ref. 18. □

Later in the paper we will show that for every pair of critical intervals R_α, R_β, where $\alpha, \beta \in \mathbb{Z}_{2m}$, there is at most one maximal VHC box $B(X,Y)$, such that $X \subseteq R_\alpha$ and $Y \subseteq R_\beta$. We will describe how given α and β we can compute the extreme points of $B(X,Y)$ and vice versa. This will allow us, instead of considering the rather complex shape of $\mathcal{X}_v$, to work with the individual VHC core inside each grid element. The latter task is much more feasible since each of the cores has a simple shape as guaranteed by the vertical and horizontal convexity property.

3.3. *Equivalence classes in $\mathcal{I}$ and finite search schedule*

In Section 1.3 we defined contamination regions in order to capture equivalence of positions of the *evader*. In this section we explore similarities between mutual positions of the *two pursuers* in order to group together equivalent visibility tuples and also the corresponding equivalent canonical information states.

We define a binary relation on the visibility tuples. Let P be a polygon with a corresponding partition of ∂P into critical intervals. Consider a pair of visibility tuples

$$I_v^i = (x_0^i, y_0^i, x_1^i, y_1^i, k), i \in \{0,1\} .$$

We say that I_v^0 is **similar** to I_v^1 if there exist critical intervals R_α, R_β, R_γ, and R_δ, such that, for $i \in \{0,1\}$, $x_0^i \in R_\alpha$, $y_0^i \in R_\beta$, $x_1^i \in R_\gamma$, $y_1^i \in R_\delta$. Clearly, "similar" is an equivalence relation so from now on we just say that I_v^0 and I_v^1 are similar. We denote the equivalence class containing I^0 and I^1 as $(\alpha, \beta, \gamma, \delta, k)$. The relation partitions $\mathcal{I}_v$ into $O(m^4)$ equivalence classes.

The "similar" relation on visibility tuples can be extended to a relation on canonical information states. Let I^0 and I^1 be two canonical information states, such that for $i \in \{0,1\}$, I^i is a concatenation of a visibility tuple I_v^i and a bits tuple I_b. We say that I^0 and I^1 are **similar** if I_v^0 and I_v^1 are similar. The "similar" relation over informations states is also an equivalence relation. It partitions the information space $\mathcal{I}$ into $O(m^4)$ equivalence classes of the form $(\alpha, \beta, \gamma, \delta, k, d_0, d_1, b_0, b_1, b_2, b_3)$, where α, β, γ, δ, and k are as described above, while d_0, d_1 and b_0, b_1, b_2, b_3, are the direction and contamination bits, correspondingly.

Lemma 4. *Let I^0 and I^1 be similar canonical information states, contained in the equivalence class v. It follows that there is a type 1 move from I^0 to I^1 entirely over canonical information states in v.*

Proof. Follows from the definition of VHC boxes. □

We define the directed **information state graph** $G = (V, E)$ to capture the equivalence classes of the similar relation over $\mathcal{I}$. The set of vertices of G is $V(G) \subset (\mathbb{Z}_{2m})^4 \times \mathbb{Z}_3 \times (\mathbb{Z}_2)^6$. A tuple $v = (\alpha, \beta, \gamma, \delta, k, d_0, d_1, b_0, b_1, b_2, b_3) \in V(G)$ if and only if v is a (non-empty) equivalence class of the similar relation over $\mathcal{I}$. The set of edges, $E(G)$, consists of all the pairs (v^0, v^1), such that there is an elementary move from some canonical information state in v^0 to some canonical information state in v^1. Intuitively, we are replacing the canonical schedules (i.e., piecewise continuous trajectories in $\mathcal{I}$) defined in Section 2.2, with corresponding finite paths in G.

Define $v \in V(G)$ to be a starting (respectively, goal) vertex if v contains a starting (respectively, goal) canonical information state. A **winning finite schedule** is a path in G from a starting to a goal vertex.

Lemma 5. *For a polygon P, there exists a winning canonical schedule, if and only if there exists a winning finite schedule.*

Proof. Let $\mathcal{I}$ and G be the canonical information space and the information state graph for the polygon P.

First, suppose there exists a winning canonical schedule, i.e., there exists a path $\pi \subset \mathcal{I}$ from a starting to a goal information state, such that π consists of elementary moves. The path π can be divided into subpaths $\pi_0, \pi_1, \ldots, \pi_k$ where each π_i is a path within the same equivalence class $v^i \subset \mathcal{I}$, $v^i \in V(G)$, $0 \leq i \leq k$. Since the path π_0 begins at a starting canonical state, then v^0 is a starting vertex. Also, for $1 \leq i \leq k$, the transition from π_{i-1} to π_i corresponds to an elementary move, so $(v^{i-1}, v^i) \in E(G)$. It follows that the corresponding path $v^0, v^1, \ldots, v^k$ is a finite schedule. Finally, π_k ends at a goal canonical information state, so v^k is a goal vertex and the path $v^0, v^1, \ldots, v^k$ represents a winning finite schedule.

Second, suppose that there exists a winning finite schedule $v^0, v^1, \ldots, v^k$. For $1 \leq j \leq k$, from $(v^{j-1}, v^j) \in E(G)$, it follows that there exist canonical information states $I''_{j-1} \in v^{j-1}$ and $I'_j \in v^j$ and also a path π_j in $\mathcal{I}$ which corresponds to an elementary move from I''_{j-1} to I'_j. Of course, in general, for $1 \leq j \leq k-1$, I'_j is different from I''_j, so simply concatenating the paths π_1, $\pi_2, \ldots \pi_k$ will not yield a valid canonical schedule. On the other hand, since for $1 \leq j \leq k-1$, I'_j and I''_j belong to the same equivalence class v^j, we can apply Lemma 4, which states that there is a (type 1) elementary move between the two states, thus there is a path ρ_j from I'_j to I''_j. Define the path $\pi \subset \mathcal{I}$ as a concatenation of $\pi_1, \rho_1, \pi_2, \ldots, \pi_{k-1}, \rho_{k-1}, \pi_k$. The vertex v^0 is a starting vertex, thus $I''_0 \in v^0$ is a starting canonical information state, therefore π is a canonical schedule. Finally, v^k is a goal vertex, thus $I'_k \in v^k$ is a goal canonical information state, so π represents a winning canonical schedule. $\square$

In Lemma 5 we showed how to transform a winning canonical schedule into a winning finite schedule, however, we have not described how to construct a winning canonical schedule, given only the polygon P. We do this in the next section.

4. Algorithm for finding a finite search schedule

We now provide an algorithm which, given P, first constructs the information state graph G and then performs a breadth-first search in G to find a winning finite schedule. If for a given polygon P and graph G such a schedule exists, it will serve as a description of a winning strategy for the two pursuers. The running time of the algorithm is $O(m^4 + m^2 n + n^2)$ where n is the number of edges and m is the number of concave regions of the polygon.

In order to construct the graph G, it is sufficient to construct the vertex set $V(G)$ and the edge set $E(G)$. In the next sections we provide details for both constructions.

4.1. *Constructing $V(G)$*

In this section, we describe how to construct $V(G)$, given the parameters of all the $O(m^2)$ VHC boxes defined by the critical points. Since the parameters of each box will be used in $O(m^2)$ different computations, it is helpful to precompute them.

Suppose $p \in \partial P$ is a critical point. By constructing the visibility polygon for p, we can identify all pairs (q^1, q^2) such that (q^1, q^2) corresponds to a (left or right) gap edge relative to p. We define the pairs in $\{(p, q^1), (p, q^2), (q^1, p), (q^2, p)\} \cap \mathcal{X}$, to be **critical gap configurations**.

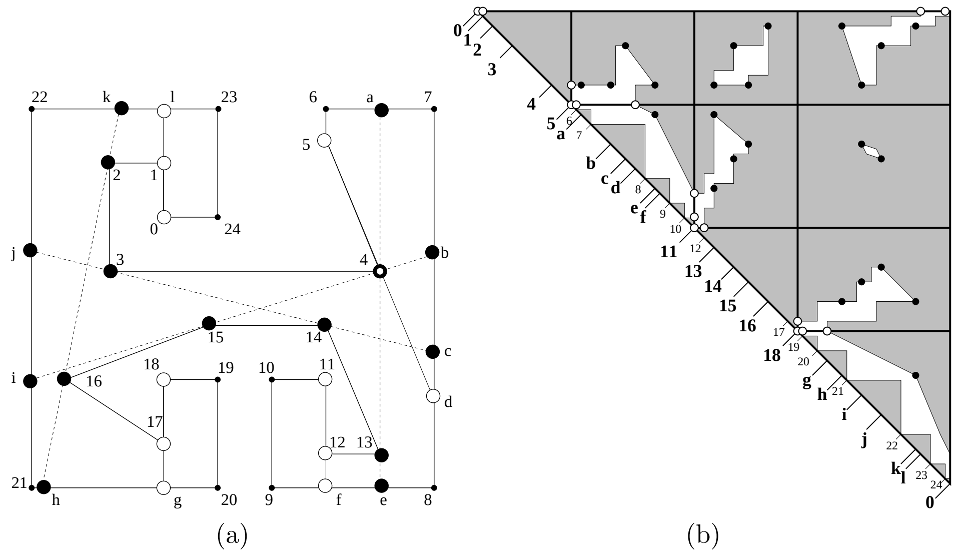


Fig. 8. Part (a) represents a polygon in which the critical points and the endpoints of the corresponding gap edges are drawn as white circles. The bitangents are drawn as dashed lines and the bitangent points are drawn as black circles. Part (b) represents the VOD of the polygon with the critical gap configurations drawn as white circles and the bitangent configurations drawn as black circles.

Let $c, d \in \partial P$ be mutually visible vertices and $x \in P$ be an interior point of the segment $\overline{cd}$. If there exist points $c', d' \in \partial P$ such that the pairs (c, c') and (d, d') form gap edges relative to x, we say that c and d define a **bitangent**. We call c, d, c' and d' **bitangent points**. For example, consider Figure 8(a) and let $c = 2$, $d = 16$, $c' = k$, $d' = h$ and let x be the midpoint of the segment $\overline{2, 16}$. Relative to x, the pairs $(2, k)$ and $(16, h)$ form a right and left gap edges, respectively, therefore 2, 16, h and k are bitangent points. If $p, q \in \{c, d, c', d'\}$ and $p \neq q$ we define the pair $\langle p, q\rangle \in \mathcal{X}$ as a **bitangent configuration**.

Lemma 6. *For a polygon with n edges and m concave regions, the parameters of all VHC boxes defined by the corresponding critical grid can be precomputed in a $2m \times 2m$ matrix in $O(n^2)$ time. Thus, given the matrix and $\alpha, \beta \in \mathbb{Z}_{2m}$, in constant time we can determine the critical intervals, R_α and R_β, and the parameters of the VHC box $B(X, Y)$, where $X \subseteq R_\alpha$ and $Y \subseteq R_\beta$, if such a box exists.*

Proof. The lemma summarizes the discussion so far: every extreme point of a VHC box is either a bitangent configuration or a critical gap configuration. By traversing the boundary, in $O(n)$ time we can construct a vector of size $2m$ which stores the critical points, i.e., the endpoints of the critical intervals $R_0, R_1, \dots, R_{2m-1}$. For each critical point in $O(n)$ time we can construct the corresponding visibility polygon which determines the critical gap configurations. So the time to determine the critical gap configurations for all critical points is $O(n + 2mn) = O(mn)$.

In order to determine the bitangent points, for every vertex p_i of P we can compute the visibility polygon of p_i in $O(n)$ time [19,20,21]. Since there are n vertices, the total time to determine the bitangent configurations is $O(n^2)$. Thus the total time to determine the parameters of all VHC boxes is $O(mn + n^2) = O(n^2)$. □

Lemma 7. *For a given polygon, if the parameters of all the boxes are precomputed, then $V(G)$, the vertex set of the information state graph G, can be constructed in time $O(m^4)$.*

Proof. Let $v = (\alpha, \beta, \gamma, \delta, k, d_0, d_1, b_0, b_1, b_2, b_3)$ be a tuple in which α, β, γ, $\delta \in \mathbb{Z}_{2m}$ are the indices of critical intervals R_α, R_β, R_γ, R_δ, while $k \in \mathbb{Z}_3$ corresponds to the order, and the d_i's and b_l's are the direction and contamination bits correspondingly. We show how to determine whether $v \in V(G)$.

If the parameters of each box are precomputed as described in Lemma 6, then in constant time we can construct VHC boxes $B(X_0, Y_0)$ and $B(X_1, Y_1)$, where $X_0 \subseteq R_\alpha$, $Y_0 \subseteq R_\beta$, $X_1 \subseteq R_\gamma$, $Y_1 \subseteq R_\delta$. If any of the two boxes does not exist, then $v \notin V(G)$, so assume that both boxes exist.

Note that $v \in V(G)$ does not depend on the values of the bits d_i and b_l, but only on the mutual position of the parameters of the boxes, as well as on the value of $k \in \mathbb{Z}_3$. Therefore, using Lemma 2 we can determine the existence of a visibility tuple $I_v = (x_0, y_0, x_1, y_1, k) \in \mathcal{I}_v$, where $(x_i, y_i) \in C(X_i, Y_i)$, $i = 0, 1$. But this is equivalent to determining whether there exists a canonical information state $I = (x_0, y_0, x_1, y_1, k, d_0, d_1, b_0, b_1, b_2, b_3) \in \mathcal{I}$ where $(x_i, y_i) \in C(X_i, Y_i)$, which itself is equivalent to determining whether $v \in V(G)$.

To construct the set $V(G)$, we consider all possible choices of α, β, δ, $\gamma \in \mathbb{Z}_{2m}$, $k \in \mathbb{Z}_3$, d_i, $b_l \in \mathbb{Z}_2$. Ignoring d_i and b_l, for each tuple I_v, we can evaluate Equation 1 from Lemma 2 in $O(1)$ time. Since the number of all such tuples is $O(m^4)$, the set $V(G)$ can be constructed in $O(m^4)$ time. □

4.2. *Constructing $E(G)$*

In order to construct the edge set, $E(G)$, of the information state graph G, we regard $E(G)$ as a (not necessarily disjoint) union of sets of edges:

$$E(G) = \bigcup_{i=0}^{5} E_i \ ,$$

where for $0 \leq i \leq 5$, $E_i \subset V(G) \times V(G)$ is set of edges, such that $(v^0, v^1) \in E_i$ if and only if there is a type i elementary move from a canonical information state in v^0 to a canonical information state in v^1.

Before we show how to construct each of the sets E_i, $0 \leq i \leq 5$, we extend to vertices the notion of independent VHC boxes from Section 3.1. Consider a vertex $v = (\alpha, \beta, \gamma, \delta, k, d_0, d_1, b_0, b_1, b_2, b_3) \in V(G)$ with corresponding VHC boxes $B(X_0, Y_0)$ and $B(X_1, Y_1)$, where $X_0 \subseteq R_\alpha$, $Y_0 \subseteq R_\beta$, $X_1 \subseteq R_\gamma$, and $Y_1 \subseteq R_\delta$. We say that v is an **independent vertex** if $B(X_0, Y_0)$ and $B(X_1, Y_1)$ are independent. Otherwise, v is a **dependent vertex**.

4.2.1. *Constructing* E_0

The edge corresponds to the completely technical type 0 move, during which we change the reference point x_0 in a tuple. As a result of the move, the order k may change. Since there is no real motion of the light segments, only a relabeling of their endpoints, the other elements in the tuple are merely permuted,

Consider vertex $v^0 = (\alpha^0, \beta^0, \gamma^0, \delta^0, k^0, d_0^0, d_1^0, b_0^0, b_1^0, b_2^0, b_3^0) \in V(G)$, with corresponding VHC box $B(X_0^0, Y_0^0)$ and $B(X_1^0, Y_1^0)$, where $X_0^0 \subseteq R_{\alpha^0}$, $Y_0^0 \subseteq R_{\beta^0}$, $X_1^0 \subseteq R_{\gamma^0}$, and $Y_1^0 \subseteq R_{\delta^0}$. Using Lemma 2, we can construct a tuple $I^0 = (x_0^0, y_0^0, x_1^0, y_1^0, k^0, d_0^0, d_1^0, b_0^0, b_1^0, b_2^0, b_3^0)$, such that $(x_j^0, y_j^0) \in B(X_j^0, Y_j^0)$, $j \in \{0, 1\}$ and k^0 corresponds to the order of x_0^0, y_0^0, x_1^0, y_1^0. As shown in Section 2.3, given I^0, in constant time we can build a second tuple $I^1 = (x_0^1, y_0^1, x_1^1, y_1^1, k^1, d_0^1, d_1^1, b_0^1, b_1^1, b_2^1, b_3^1)$, which is the result of applying a type 0 move on I^0. From I^1 in constant time we can build $v^1 = (\alpha^1, \beta^1, \gamma^1, \delta^1, k^1, d_0^1, d_1^1, b_0^1, b_1^1, b_2^1, b_3^1) \in V(G)$, where $x_0^1 \subseteq R_{\alpha^1}$, $y_0^1 \subseteq R_{\beta^1}$, $x_1^1 \subseteq R_{\gamma^1}$, and $y_1^1 \subseteq R_{\delta^1}$.

In order to compute all the edges in E_0, we need to consider two cases. There are $O(m^4)$ independent vertices v^0 and from each of them we can compute v^1 in time $O(1)$. So the total time for constructing the E_0 edges that start from an independent vertex is $O(m^4)$. On the other hand, there are $O(m^2)$ dependent vertices v^0 and from each of them we can compute v^1 in time $O(n)$. So the total time for constructing the E_0 edges that start from a dependent vertex is $O(nm^2)$. Summing up, the total time to construct E_0 is $O(m^4 + nm^2)$.

4.2.2. *Constructing* E_1

Intuitively, the edges in E_1 represent type 1 elementary moves during which pursuer 0 moves continuously, while pursuer 1 is stationary. There are no jumps and the relative order of the endpoints of the light segments does not change. We are not interested in the type 1 moves in which pursuer 0 stays within the same grid element. Instead, we consider the moves in which pursuer 0 crosses between two *neighboring* grid elements.

Consider distinct vertices $v^i = (\alpha^i, \beta^i, \gamma, \delta, k, d_0, d_1, b_0, b_1, b_2, b_3) \in V(G)$, $i =$

$0, 1$, and the corresponding VHC boxes $B(X_0^0, Y_0^0)$, $B(X_0^1, Y_0^1)$ and $B(X_1, Y_1)$, where $X_0^0 \subseteq R_{\alpha^0}$, $Y_0^0 \subseteq R_{\beta^0}$, $X_0^1 \subseteq R_{\alpha^1}$, $Y_0^1 \subseteq R_{\beta^1}$, $X_1 \subseteq R_\gamma$, and $Y_1 \subseteq R_\delta$. (To avoid tedious technicalities, i.e., a change in k, assume also that $\alpha^0 \neq \beta^1$.) If $B(X_0^0, Y_0^0)$ and $B(X_0^1, Y_0^1)$ are not boxes from neighboring grid elements, then $(v^0, v^1) \notin E_1$. Therefore, we do not have to consider all possible pairs of vertices $v^0, v^1 \in V(G)$. Instead, for each of the $O(m^4)$ possible choices of $v^0 \in V(G)$, there are $O(1)$ possible choices for v^1 which satisfy the "neighboring grid element" condition. So the total of pairs (v^0, v^1) which we have to consider as condidates for E_1 is $O(m^4)$.

Denote by $\overline{s_1, s_4}$ the (vertical or a horizontal) grid segment that $B(X_0^0, Y_0^0)$ and $B(X_0^1, Y_0^1)$ share. If $\overline{s_1, s_4} \cap \mathcal{X}_v = \emptyset$, then $(v^0, v^1) \notin E_1$. So assume that $\overline{s_1, s_4} \cap \mathcal{X}_v \neq \emptyset$ and let $\overline{s_2, s_3}$ be the maximal subsegment of $\overline{s_1, s_4}$ whose interior lies entirely in $\mathcal{X}_v$. There exists a sufficiently small VHC box $B(X, Y)$ whose interior contains $\overline{s_2, s_3}$.

We apply Lemma 2 for k, $B(X, Y)$ and $B(X_1, Y_1)$ as defined above. If Condition 1 from the lemma is not satisfied, then $(v^0, v^1) \notin E_1$. On the other hand, suppose that the condition is satisfied by some $I_v = (x_0, y_0, x_1, y_1, k) \in \mathcal{I}_v$. From the definition of $B(X, Y)$, it follows that $I_v \in \mathcal{I}_v$ if and only if there are points $(x_0^i, y_0^i) \in B(X_0^i, Y_0^i)$, sufficiently close to (x_0, y_0), so that $I_v^i = (x_0^i, y_0^i, x_1, y_1, k) \in \mathcal{I}_v$, $i = 0, 1$. This is equivalent to the existence of a type 1 move between canonical information states $I^i = (x_0^i, y_0^i, x_1, y_1, k, d_0, d_1, b_0, b_1, b_2, b_3) \in \mathcal{I}$, $i = 0, 1$, i.e., equivalent to $(v^0, v^1) \in E_1$.

It follows that determining whether $(v_0, v_1) \in E_1$ is equivalent to evaluating Condition 1 from Lemma 2. The condition can be verified and I_v can be constructed in constant time for independent vertices v^0 and v^1 and in time $O(n)$ for dependent ones. There are $O(m^4)$ different independent boxes and $O(m^2)$ dependent ones, so to construct the edges in E_1 we need time $O(m^4 + nm^2)$.

4.2.3. *Constructing E_2*

Intuitively, a type 2 edge represents a moment during which pursuer 0 clears the corner corresponding to a non-reflex vertex while pursuer 1 is stationary.

Consider $v^i = (\alpha, \beta, \gamma, \delta, k, d_0, d_1, b_0^i, b_1^i, b_2^i, b_3^i) \in V(G)$, $i = 0, 1$ with corresponding boxes $B(X_0, Y_0)$ and $B(X_1, Y_1)$, where $X_0 \subseteq R_\alpha$, $Y_0 \subseteq R_\beta$, $X_1 \subseteq R_\gamma$, $Y_1 \subseteq R_\delta$. If neither of $B(X_0, Y_0)$ or $B(X_1, Y_1)$ contains a non-reflex vertex, then $(v^0, v^1) \notin E_2$. So assume that there is a non-reflex vertex $p \in \partial P$ which belongs to one of the boxes. Let $p^-, p^+ \in \partial P$, $p \in \partial P(p^-, p^+)$ and also define the intervals $X = \partial P(p^-, p)$ and $Y = \partial P(p, p^+)$. Let $\partial P(p^-, p^+)$ be sufficiently small so that $B(X, Y) = C(X, Y) \subseteq \mathcal{X}_v$ is a VHC core. Without loss of generality, possibly after some relabeling, we can assume that $p \in B(X_1, Y_1)$ and neither of X or Y overlaps with X_0 or Y_0. Suppose that (x_0, y_0) is an arbitrary point from $C(X_0, Y_0)$. If pursuer 0 is stationary at (x_0, y_0) while pursuer 1 converges from (p^-, p^+) to (p, p), this corresponds to a type 2 move, as described in Section 2.3.

Given $B(X, Y)$ and $B(X_1, Y_1)$, in constant time we can construct $I^i =$

$(x_0, y_0, p^-, p^+, k, d_0, d_1, b_0^i, b_1^i, b_2^i, b_3^i)$, $i = 0, 1$. Using I^0 and I^1 in constant time we can determine whether $(v^0, v^1) \in E_2$ by just verifying that the contamination bits b_0^i, b_1^i, b_2^i, b_3^i, $i = 0, 1$, are consistent with the bit changes for a type 2 move between I^0 and I^1.

In order to construct the entire E_2, we consider every pair of critical intervals $R_\alpha, R_\beta \in \mathbb{Z}_{2m}$, every reflex vertex $p_i \in \mathbb{Z}_n$, as well all the possible choces for k, b_0, b_1, $d_0, \ldots d_3$. Given those, there is a unique choice of $B(X, Y)$ and $B(X_0, Y_0)$, and therefore of (v^0, v^1) as described above. Since membership $(v^0, v^1) \in E_2$ can be decided in constant time, it follows that the total time to construct E_2 is $O(nm^2)$.

4.2.4. *Constructing* E_3

Intuitively, an edge in E_3 corresponds to a type 3 move between canonical information states. The motion of each of the pursuers is continuous, all within the corresponding VHC core. The order k changes and there is a possible change in the contamination bits.

Consider $v^i = (\alpha, \beta, \gamma, \delta, k, d_0, d_1, b_0^i, b_1^i, b_2^i, b_3^i) \in V(G)$, $i = 0, 1$ with corresponding boxes $B(X_0, Y_0)$ and $B(X_1, Y_1)$, where $X_0 \subseteq R_\alpha$, $Y_0 \subseteq R_\beta$, $X_1 \subseteq R_\gamma$, $Y_1 \subseteq R_\delta$. If none of X_0 or Y_0 intersects with X_1 or Y_1, then there can be no overlap between the endpoints of the light segments, thus there can be no type 3 elementary move and $(v^0, v^1) \notin E_3$. So without loss of generality, possibly after some relabeling, we can assume that $Y_0 \cap X_1 \neq \emptyset$. Choose $x_0, y_0, y_1 \in \partial P$, such that $y_0 \in Y_0 \cap X_1$, also $(x_0, y_0) \in C(X_0, Y_0)$, $(y_0, y_1) \in C(X_1, Y_1)$. Since $Y_0 \cap X_1$ is a nonempty open set there exists a sufficiently small interval $\partial P(x_1^1, x_1^0) \subset Y_0 \cap X_1$, with the property that $y_0 \in \partial P(x_1^1, x_1^0)$ and also that the entire segment between the points (x_1^1, y_1) and (x_1^0, y_1) lies in $C(X_1, Y_1)$. If pursuer 0 is stationary at (x_0, y_0), while pursuer 1 moves from (x_1^0, y_1) over (y^0, y_1) to (x_1^1, y_1), this corresponds exactly to a type 3 move, as described in Section 2.3.

To determine whether $(v^0, v^1) \in E_3$, we just have to verify that the contamination bits b_0^i, b_1^i, b_2^i, b_3^i, $i = 0, 1$, are consistent with the bit changes for a type 3 move. More precisely, let $I^i = (x_0, y_0, x_1^i, y_1, k, d_0, d_1, b_0^i, b_1^i, b_2^i, b_3^i)$, $i = 0, 1$. In constant time we can determine whether there is a type 3 move between I^0 and I^1. If given α, β, γ, δ, k and the bits, then v^0 and v^1 can be constructed and membership in E_3 can be determined in constant time. There are $O(m^4)$ possible values for α, β, γ, δ, k and the bits, so the total time to construct E_3 is $O(m^4)$.

4.2.5. *Constructing* E_4 *and* E_5

Intuitively, the edges in E_4 and E_5 correspond to the type 4 and 5 jump elementary moves. For every elementary move, given the original canonical information state I^0 and the type of the move we can construct the resulting canonical information state I^1. Thus our main goal is to determine whether there is a feasible jump between the corresponding visibility tuples I_v^0 and I_v^1. If the latter jump exists, we can determine in constant time whether there is a type 4 or type 5 jump between the corresponding information states I^0 and I^1.

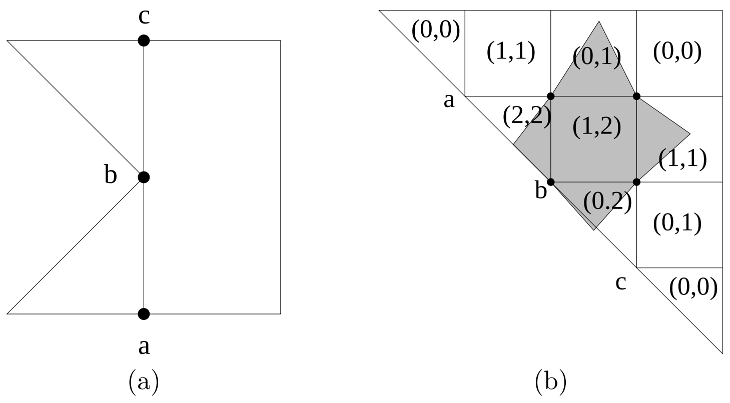


Fig. 9. (a) Tangent defined by points a, b and c. (b) Corresponding configurations in $\mathcal{X}$. The points in $\mathcal{X}_v$ are white, the points in $\mathcal{X} - \mathcal{X}_v$ are grey.

Consider a tangent defined by the points $a, b, c \in \partial P$, see Figure 9(a). The tangent induces two corresponding jumps in X: a jump from (a, b) to (a, c) is a jump over a left gap edge represented a horizontal segment and a jump from (c, b) to (c, a) is a jump over a right gap edge, represented as a vertical segment, see Figure 9(b). Just as in the definition of the type 4 and type 5 elementary moves, we will only discuss the jumps over left gap edges. The jumps over right gap edges are analogous, e.g., they can be constructed by considering a mirror image of the polygon.

Suppose pursuer 0 performs a jump over a gap edge from (a, b) to (a, c), such that $x_0 = a$, $y_0^0 = b$ and $y_0^1 = c$. The points $a, b, c \in \partial P$ induce a partition of $\mathcal{X}_v$ into six regions (k^0, k^1), $0 \le k^0 \le k^1 \le 2$, where every region (k^0, k^1) contains all the points (x^1, y^1) such that there exist $I_v^i = (x_0, y_0^i, x_1, y_1, k^i)$, $i = 0, 1$. Intuitively, the visibility tuples I_v^0 and I_v^1 denote a move in which pursuer 0 is making a jump from x_0, y_0^0 to x_0, y_0^1, while pursuer 1 is stationary at (x_1, y_1). Region (k^0, k^1) denotes all the positions of pursuer 1, such that the order of the I_v^0 is k^0 and the order I_v^1 is k^1, see Figure 9(b).

Clearly, if we fix x_0, y_0^0 and y_0^1 and we are also given the VHC box of (x_1, y_1), in constant time we can determine the feasibility of a jump. However, since the number of possible positions for x_0, y_0^0 and y_0^1 is infinite, our next goal is to partition all the jump moves into a finite number of classes.

Consider critical intervals R_α, R_{β^0}, R_{β^1} and let $B(X_0, Y_0^0)$ and $B(X_0, Y_0^1)$ be the corresponding VHC boxes where $X_0 \subseteq R_\alpha$, $Y_0^0 \subseteq R_{\beta^0}$, $Y_0^1 \subseteq R_{\beta^1}$. Let $(x_0', x_0'') \subseteq X_0$ be a maximal interval with the property that for every $x^0 \in (x_0', x_0'')$, there is a jump from $(x^0, y^0) \in C(X_0^0, Y_0^0)$ to $(x^0, y^1) \in C(X_0^1, Y_0^1)$. We define (x_0', x_0'') to be a **jump interval** and we use it to group together equivalent jumps from $B(X_0^0, Y_0^0)$ to $B(X_0^1, Y_0^1)$. For two given boxes as defined above a jump interval (x_0', x_0'') may not exist or may not be unique, yet its boundaries always correspond to either bitangent or critical gap configurations. Therefore, since each such point can border at most two intervals and the number of both the bitangent configurations and the critical gap configurations is $O(m^2)$, it follows that the number of jump intervals is also $O(m^2)$.

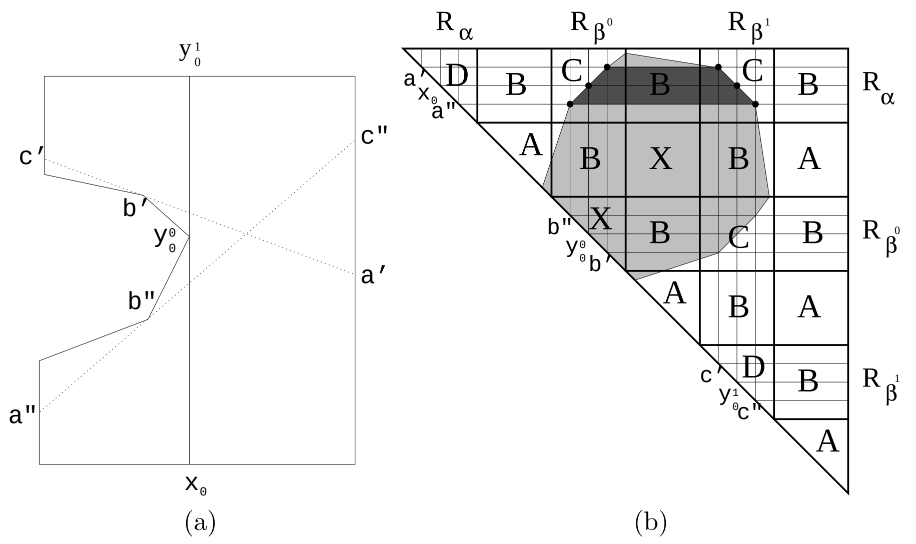


Fig. 10. Jump interval and partition of $\mathcal{X}$ into regions of different labels.

Let $v^i = (\alpha, \beta^i, \gamma, \delta, k^i, d_0^i, d_1^i, b_0^i, b_1^i, b_2^i, b_3^i)$, $i = 0, 1$ be vertices in $V(G)$ with corresponding boxes $B(X_0, Y_0^0)$, $B(X_0, Y_0^1)$ and $B(X_1, Y_1)$, where $X_0 \subseteq R_\alpha$, $Y_0^0 \subseteq R_{\beta^0}$, $Y_0^1 \subseteq R_{\beta^1}$, $X_1 \subseteq R_\gamma$, $Y_1 \subseteq R_\delta$. Assume that (a', a'') is a jump interval for boxes $B(X_0, Y_0^0)$ and $B(X_0, Y_0^1)$. Suppose that b', b'', c' and c'' are such that there is a jump from (a', b') to (a', c') and also a jump from (a'', b'') to (a'', c'') and let $x_0 \in (a', a'')$, $y_0^0 \in (b'', b')$ and $y_0^1 \in (c', c'')$ by such that there is a jump from (x_0, y_0^0) to (x_0, y_0^1) as shown in Figure 10(a). Figure 10(b) illustrates the corresponding VOD $\mathcal{X}_v$. The white points correspond to the points in $\mathcal{X}_v$. The shaded points are the ones in $\mathcal{X} - \mathcal{X}_v$. The trapezoid which is shaded darker represents all the different jumps for the given jump interval. The lines corresponding to the boundaries of the intervals R_α, R_{β^0} and R_{β^1}, shown as thicker lines in Figure 10(b), are part of the grid and divide $\mathcal{X}$ into regions labeled with X, A, B, C and D. Note that there are no points from $\mathcal{X}_v$ in a region labeled X, so $B(X_1, Y_1)$ has to be in one of the other four. If $B(X_1, Y_1)$ lies in a region labeled A or B, then we can determine in constant time whether there exist I_v^0 and I_v^1 and a corresponding type 4 (resp. type 5) move between them, and we can determine in constant time whether $(v^0, v^1) \in E_4$ (resp. $\in E_5$). On the other hand, if $B(X_1, Y_1)$ lies in a region labeled C or D, we need to construct a finite number of visibility polygons (similar to the proof of Lemma 2) to determine membership in E_4 and E_5.

What is the time needed to construct E_4 and E_5? For a fixed jump interval the intervals R_β, R_{β^0} and R_{β^1} are fixed as well. There are $O(m^2)$ possible choices for R_γ and R_δ. Since there are $O(m^2)$ boxes $B(X_1, Y_1)$ which lie in regions labeled A or B and determining the existence of an edge for each one takes $O(1)$ time, the total time for those is $O(m^2)$. On the other hand, there are $O(1)$ boxes which lie in a region labeled C or D and each one takes $O(n)$ time, so the total time for the C and D regions is $O(n)$. Thus, for a single jump interval the time is $O(m^2 + n)$. There are $O(m^2)$ jump intervals so the total time to construct E_4 and E_5 is $O(m^2(m^2 + n)) = O(m^4 + nm^2)$.

4.3. *Finding a winning finite schedule*

We can precompute the bitangent and critical gap configurations in $O(n^2)$ time. Given this precomputation, we can construct the graph G in time $O(m^4 + m^2 n)$. Finally, we can run breadth-first search in the graph to find a winning finite schedule. The size of the vertex set, $V(G)$, is $O(m^4)$. The size of the edge sets E_0, E_1, E_2 and E_3 is also $O(m^4)$ since every vertex in $V(G)$ has a constant outdegree for each of those edge sets. Finally, the size of the edge sets E_4 and E_5 is $O(m^4)$ as well, since there is a constant number of edges for every jump interval and box $B(X_1, Y_1)$. Thus, the size of $E(G)$ is $O(m^4)$; therefore, bread-first search in G will take $O(m^4)$ time.

It follows that determining the existence and constructing a winning finite schedule can be done in time $O(n^2 + nm^2 + m^4)$.

5. Conclusion

We presented a complete algorithm for a pair of pursuers, each with one rotating flashlight, searching for an moving target in a simple polygon. For a polygon with n edges and m concave regions, the algorithm in time $O(n^2 + nm^2 + m^4)$ decides whether it can be cleared by the pursuers, and if so, constructs a search schedule. The algorithm can be implemented and embedded on any moving devices with unidirectional vision (flashlights, lasers, or cameras). A natural direction for extending the current results is designing a similar algorithm for two pursuers with 360° vision. A more ambitious goal is to provide an algorithm for searching a polygon without holes using any number of pursuers. Another interesting problem is combining the results of our paper with the minimal sensing approach of Sachs et al,[22] i.e., whether the two pursuers can find a winning strategy without prior knowledge of the shape of the polygon.

References

1. Steven M. LaValle and John E. Hinrichsen. Visibility-based pursuit-evasion: the case of curved environments. *IEEE Transactions on Robotics and Automation*, 17(2):196–202, April 2001.
2. Ichiro Suzuki and Masafumi Yamashita. Searching for a mobile intruder in a polygonal region. *SIAM Journal on Computing*, 21(5):863–888, October 1992.
3. Christian Icking and Rolf Klein. The two guards problem. *International Journal of Computational Geometry and Applications*, 2(3):257–285, 1992.
4. Paul J. Heffernan. An optimal algorithm for the two-guard problem. *International Journal of Computational Geometry and Applications*, 6(1):15–44, 1996.
5. L. H. Tseng, Paul J. Heffernan, and D. T. Lee. Two-guard walkability of simple polygons. *International Journal of Computational Geometry and Applications*, 8(1):85–116, February 1998.
6. D. Crass, Ichiro Suzuki, and Masafumi 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.
7. Jae-Ha Lee, Sang-Min Park, and Kyung-Yong Chwa. Searching a polygonal room with a door by a 1-searcher. *International Journal of Computational Geometry and Applications*, 10(2):201–220, April 2000.
8. Jae-Ha Lee, Sung Yong Shin, and Kyung-Yong 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.
9. Borislav H. Simov, Giora Slutzki, and Steven M. LaValle. Pursuit-evasion using beam detection. In *Proceedings, IEEE International Conference on Robotics and Automation (ICRA)*, pages 1657–1662, San Francisco, CA, USA, April 2000.
10. Steven M. LaValle, Borislav H. Simov, and Giora Slutzki. An algorithm for searching a polygonal region with a flashlight. In *Proceedings, ACM Symposium on Computational Geometry (SCG)*, pages 260–269, Hong-Kong, June 2000.
11. Sang-Min Park, Jae-Ha Lee, and Kyung-Yong Chwa. Visibility-based pursuit-evasion in a polygonal region by a searcher. In F. Orejas, P. G. Spirakis, and J. Leeuwen, editors, *Proceedings of 28th ICALP*, volume 2076 of *Lecture Notes in Computer Science*, pages 456–468. Springer-Verlag, Berlin, 2001.
12. Leonidas J. Guibas, Jean-Claude Latombe, Steven M. LaValle, David Lin, and Rajeev 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*, volume 1272 of *Lecture Notes in Computer Science*, pages 17–30. Springer-Verlag, Berlin, 1997.
13. Alon Efrat, Leonidas J. Guibas, Sariel Har-Peled, David C. Lin, Joseph S. B. Mitchell, and T. M. Murali. Sweeping simple polygons with a chain of guards. In *Proceedings, ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 927–936, San Francisco, CA, USA, 2000.
14. Ichiro Suzuki, Yuichi Tazoe, Masafumi Yamashita, and Tiko Kameda. Searching a polygonal region from the boundary. *International Journal of Computational Geometry and Applications*, 11(5):529–553, October 2001.
15. Ichiro Suzuki, Masafumi Yamashita, Hideki Umemoto, and Tsunehiko Kameda. Bushiness and a tight worst-case upper bound on the search number of a simple polygon. *Information Processing Letters*, 66(1):49–52, April 1998.
16. Masafumi Yamashita, Hideki Umemoto, Ichiro Suzuki, and Tsunehiko Kameda. Searching for mobile intruders in a polygonal region by a group of mobile searchers. *Algorithmica*, 31(2):208–236, 2001.

17. Rene Vidal, Omid Shakernia, H. Jin Kim, David Hyunchul Shim, and Shankar Sastry. Probabilistic pursuit-evasion games: Theory, implementation and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, October 2002.
18. Steven M. LaValle, Borislav H. Simov, and Giora Slutzki. An algorithm for searching a polygonal region with a flashlight. *International Journal of Computational Geometry and Applications*, 12(1 & 2):87–113, February & April 2002.
19. Hossam A. ElGindy and David Avis. A linear algorithm for computing the visibility polygon from a point. *Journal of Algorithms*, 2(2):186–197, 1981.
20. Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete Computational Geometry*, 6(5):485–524, 1991.
21. Leonidas Guibas, John Hershberger, Daniel Leven, Micha Sharir, and Robert E. Tarjan. Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons. *Algorithmica*, 2(1):209–233, March 1987.
22. Shai Sachs, Steven M. LaValle, and Stjepan Rajko. Visibility-based pursuit-evasion in an unknown planar environment. *International Journal of Robotics Research*, 23(1):3–26, 2004.

Appendix A. Proof of Lemma 2

Assume that X_0, Y_0, X_1, Y_1 and k are as defined in Lemma 2. Also, for convenience, assume that we order the points of ∂P starting from $\underline{x_0}$, the left boundary of the interval X_0. We breakdown the proof according to the different values of k, where $0 \leq k \leq 2$:

A.1. *Proof of Lemma 2 for the case $k = 0$*

We prove that if $k = 0$, then Condition 1 is satisfied if and only if $\underline{x_0} \prec \underline{y_0} \prec \overline{x_1}$.

Proof.
($\Leftarrow$) For the sake of contradiction, assume that $\underline{x_0} \prec \overline{x_1} \preceq \underline{y_0}$. It follows that for every $(x_i, y_i) \in B(X_i, Y_i)$, $i = 0, 1$:

$$\underline{x_0} \preceq x_1 \preceq \overline{x_1} \preceq \underline{y_0} \prec y_0 \ ,$$

so if $\underline{x_0} \prec x_1 \preceq y_0$, then Condition 1 cannot be satisfied.

($\Rightarrow$) Suppose that $\underline{x_0} \prec \underline{y_0} \prec \overline{x_1}$. Let $(x_0^*, \underline{y_0})$ and $(\overline{x_1}, y_1^*)$ be two extreme points of $C(X_0, Y_0)$ and $C(X_1, Y_1)$, correspondingly. Since,

$$\underline{x_0} \prec x_0^* \prec \underline{y_0} \prec \overline{x_1} \prec y_1^*$$

the tuple $I_v = (x_0^*, \underline{y_0}, \overline{x_1}, y_1^*, 0)$ satisfies Condition 1. □

Both the check $\underline{x_0} \prec \underline{y_0} \prec \overline{x_1}$ and the construction of I_v can be done in $O(1)$ time, regardless of the whether the boxes are independent or not.

A.2. *Proof of Lemma 2 for the case $k = 1$*

We consider three cases, depending on the mutual position of the boxes.

A.2.1. *Case with $k = 1$, $X_0 \cap X_1 \neq \emptyset$ and $Y_0 \cap Y_1 \neq \emptyset$*

We prove that if $k = 1$, $X_0 \cap X_1 \neq \emptyset$ and $Y_0 \cap Y_1 \neq \emptyset$, then Condition 1 is always satisfied.

Proof. Let $X = X_0 \cap X_1 \neq \emptyset$ and $Y = Y_0 \cap Y_1 \neq \emptyset$. From Lemma 3.9 of Ref. 18 it follows that $B(X,Y)$ is a VHC box. Consider an interior point $(x_0^*, y_0^*) \in C(X,Y) \subseteq C(X_0,Y_0)$. Suppose we move in $\mathcal{X}$ from point (x_0^*, y_0^*) down and to the right to point (x_1^*, y_1^*). Assume that the move is sufficiently short, so that $(x_1^*, y_1^*) \in C(X,Y) \subseteq C(X_1,Y_1)$. From the direction of the move it follows that $\underline{x_0} \prec x_0^* \prec x_1^* \prec y_0^* \prec y_1^*$, so Condition 1 is always satisfied by tuple $I_v = (x_0^*, y_0^*, x_1^*, y_1^*, 1)$. $\square$

Determining whether Condition 1 can be satisfied, or equivalently, whether sets X and Y are empty can be done in $O(1)$ time. In order to construct I_v, we compute the parameters of $B(X,Y)$ by computing a finite number of visibility polygons in $O(n)$ time. Given the parameters of $B(X,Y)$ we can construct I_v in $O(1)$ time.

A.2.2. *Case with $k = 1$, $X_0 \cap (X_1 \cup Y_1) = \emptyset$, $Y_0 \cap X_1 \neq \emptyset$ and $Y_0 \cap Y_1 \neq \emptyset$*

We prove that if $k = 1$, $X_0 \cap (X_1 \cup Y_1) = \emptyset$, $Y_0 \cap X_1 \neq \emptyset$ and $Y_0 \cap Y_1 \neq \emptyset$, then Condition 1 is always satisfied.

Proof. Pick $y_1^* \in Y_0 \cap Y_1$ and construct a point $(x_1^*, y_1^*) \in C(X_1, Y_1)$. Pick $y_0^* \in \partial P$ such that $x_1^* \prec y_0^* \prec y_1^*$ and $\underline{y_0} \prec y_0^* \prec y_1^*$. Construct $(x_0^*, y_0^*) \in C(X_0, Y_0)$. Since

$$\underline{x_0} \prec x_0^* \preceq \overline{x_0} \prec \underline{x_1} \preceq x_1^* \prec y_0^* \prec y_1^*$$

the tuple $I_v = (x_0^*, y_0^*, x_1^*, y_1^*, 1)$ satisfies Condition 1. $\square$

In order to construct (x_0^*, y_0^*) and (x_1^*, y_1^*) we need to construct two visibility polygons. So I_v can be constructed in time $O(n)$.

A.2.3. *Case with $k = 1$, $X_0 \cap (X_1 \cup Y_1) = \emptyset$ and $Y_0 \cap Y_1 = \emptyset$*

We prove that if $k = 1$, $X_0 \cap (X_1 \cup Y_1) = \emptyset$ and $Y_0 \cap Y_1 = \emptyset$, then Condition 1 is satisfied if and only if $\underline{x_0} \prec \underline{x_1} \prec \overline{y_0}$.

Proof.
($\Leftarrow$) For the sake of contradiction, assume that $\underline{x_0} \prec \overline{y_0} \preceq \underline{x_1}$. It follows that for every $(x_i, y_i) \in B(X_i, Y_i)$, $i = 0, 1$:

$$\underline{x_0} \prec y_0 \preceq \overline{y_0} \preceq \underline{x_1} \preceq x_1 \ ,$$

so if $\underline{x_0} \preceq y_0 \preceq x_1$, then Condition 1 cannot be satisfied.

($\Rightarrow$) Suppose that $\underline{x_0} \preceq \underline{x_1} \prec \overline{y_0}$. Let $(x_0^*, \overline{y_0})$ and $(\underline{x_1}, y_1^*)$ be two extreme points of $C(X_0, Y_0)$ and $C(X_1, Y_1)$, correspondingly. Since,

$$\underline{x_0} \prec x_0^* \preceq \overline{x_0} \prec \underline{x_1} \prec \overline{y_0} \prec \underline{y_1} \prec y_1^*$$

the tuple $I_v = (x_0^*, \overline{y_0}, \underline{x_1}, y_1^*, 1)$ satisfies Condition 1. $\square$

Both the check $\underline{x_0} \prec \underline{x_1} \prec \overline{y_0}$ and the construction of I_v can be done in $O(1)$ time, regardless of the whether the boxes are independent or not.

A.3. *Proof of Lemma 2 for the case $k = 2$*

We consider two cases, depending on the mutual position of the boxes.

A.3.1. *Case with $k = 2$, $X_0 \cap X_1 \neq \emptyset$ and $Y_0 \cap Y_1 \neq \emptyset$*

We prove that if $k = 2$, $X_0 \cap X_1 \neq \emptyset$ and $Y_0 \cap Y_1 \neq \emptyset$, then Condition 1 is always satisfied.

Proof. Let $X = X_0 \cap X_1 \neq \emptyset$ and $Y = Y_0 \cap Y_1 \neq \emptyset$. From Lemma 3.9 of Ref. 18 it follows that $B(X,Y)$ is a VHC box. Consider an interior point $(x_0^*, y_0^*) \in C(X,Y) \subseteq C(X_0,Y_0)$. Suppose we move in $\mathcal{X}$ from point (x_0^*, y_0^*) down and to the left to point (x_1^*, y_1^*). Assume that the move is sufficiently short, so that $(x_1^*, y_1^*) \in C(X,Y) \subseteq C(X_1,Y_1)$. From the direction of the move it follows that $x_0^* \prec x_1^* \prec y_1^* \prec y_0^*$, so Condition 1 is always satisfied by tuple $I_v = (x_0^*, y_0^*, x_1^*, y_1^*, 2)$. $\square$

Determining whether Condition 1 can be satisfied, or equivalently, whether sets X and Y are empty can be done in $O(1)$ time. In order to construct I_v, we compute the parameters of $B(X,Y)$ by computing a finite number of visibility polygons in $O(n)$ time. Given the parameters of $B(X,Y)$ we can construct I_v in $O(1)$ time.

A.3.2. *Case with $k = 2$, $X_0 \cap X_1 = \emptyset$*

We prove that if $k = 2$ and $X_0 \cap X_1 = \emptyset$, then Condition 1 is satisfied if and only if $\underline{\mathbf{x_0}} \prec \underline{\mathbf{y_1}} \prec \overline{\mathbf{y_0}}$.

Proof.
($\Leftarrow$) Suppose that $\underline{x_0} \prec \overline{y_0} \preceq \underline{y_1}$. It follows that for every $(x_i, y_i) \in B(X_i, Y_i)$, $i = 0, 1$:

$$\underline{x_0} \prec y_0 \preceq \overline{y_0} \preceq \underline{y_1} \preceq y_1 \ ,$$

so if $\underline{x_0} \prec y_0 \preceq y_1$, then Condition 1 cannot be satisfied.

($\Rightarrow$) Suppose that $\underline{x_0} \prec \underline{y_1} \prec \overline{y_0}$. Let $(x_0^*, \overline{y_0})$ and $(x_1^*, \underline{y_1})$ be two extreme points of $C(X_0,Y_0)$ and $C(X_1,Y_1)$, correspondingly. Since,

$$\underline{x_0} \prec x_0^* \preceq \overline{x_0} \prec \underline{x_1} \preceq x_1^* \prec \underline{y_1} \prec \overline{y_0}$$

the tuple $I_v = (x_0^*, \overline{y_0}, x_1^*, \underline{y_1}, 2)$ satisfies Condition 1. $\square$

Both the check $\underline{x_0} \prec \underline{y_1} \prec \overline{y_0}$ and the construction of I_v can be done in $O(1)$ time, regardless of the whether the boxes are independent or not.