(1)

Searching for a Mobile Intruder in a Corridor—The Open Edge Variant of the Polygon Search Problem*

David Crass
Department of Electrical Engineering and Computer Science
University of Wisconsin - Milwaukee
P.O. Box 784, Milwaukee, WI 53201, U.S.A.
davidc@csd4.csd.uwm.edu

Ichiro Suzuki
Department of Electrical Engineering and Computer Science
University of Wisconsin - Milwaukee
P.O. Box 784, Milwaukee, WI 53201, U.S.A.
suzuki@cs.uwm.edu

Masafumi Yamashita
Department of Electrical Engineering
Faculty of Engineering
Hiroshima University
Kagamiyama, Higashi-Hiroshima 724, Japan
mak@se.hiroshima-u.ac.jp

To appear in
International Journal of Computational Geometry and Applications
1994

2 cpy

Abstract The polygon search problem is the problem of searching for mobile intruders in a simple polygon by a single mobile searcher having various degrees of visibility. This paper considers the "open edge" variant of the problem in which the given polygon P must be searched without allowing undetected intruders to reach a given edge u, under an additional assumption that any number of intruders can leave and enter P through another edge v at any time. One may view P as representing a corridor with two open exits u and v, and the task of the searcher is to force all the intruders out of P through v (but not u). We present a simple necessary condition for a polygon to be searchable in this manner by the searcher having a light bulb, and then show that the same condition is sufficient for the polygon to be searchable by the searcher having two flashlights. The time complexity of generating a search schedule is also discussed.

*This work was supported in part by the National Science Foundation under grants CCR-9004346 and IRI-9307506, the Office of Naval Research under grant N00014-94-1-0284, and a Scientific Research Grant-in-Aid 03650305 from the Ministry of Education, Science and Culture of Japan.

Key words: geometry, visibility

1 Introduction

Problems related to visibility inside a simple polygon have been the subject of many recent papers. Of particular interest to us among these problems is the watchman route problem [2] [3], which is an interesting variation of the well-known art gallery problem of stationing guards in a simple polygon so that every point in the interior of the polygon will be visible from at least one guard [4] [8]. The goal of the watchman route problem is to construct a shortest tour within a given simple polygon so that every point in the interior of the polygon will be visible from at least one point on the tour. Note that this goal can be interpreted as (constructing a path for) finding *stationary* intruders located in the polygon by a single mobile searcher.

Detection of *mobile* intruders in a simple polygon was first considered in the searchlight scheduling problem [9] in which the rays of stationary searchlights are used to find the intruder. The use of a *mobile* searcher having various degrees of visibility for detecting mobile intruders was then considered as the polygon search problem in [11] where a number of necessary conditions and sufficient conditions for the given polygon to be searchable by various searchers are presented. The goal of this paper is to discuss an interesting variant of the polygon search problem.

We adopt the following formalism given in [11]. Both the searcher and the intruders are represented as a point that can move continuously within the given polygon P, and the intruders are assumed to be able to move arbitrarily faster than the searcher. For each integer $k \geq 1$, the k-searcher is the searcher having k flashlights whose visibility is limited to k rays emanating from his position, where the directions of the rays can be changed continuously with bounded angular rotation speed. The ∞-searcher is the searcher having a light bulb who can see in all directions simultaneously at any time. (A searchlight used in [9] is equivalent to a stationary 1-searcher.) We say that a point or an intruder is *illuminated* at the given time if either (1) it is hit by one of the rays of the k-searcher, or (2) it is visible from the position of the ∞-searcher. A *schedule* of the k-searcher is a sequence of the following *elementary actions*:

1. Aim a flashlight at the given point.
2. Rotate a flashlight either clockwise or counterclockwise to illuminate the given point.
3. Move over a segment, aiming each flashlight either in the given fixed direction, or at or through the given point.

A *schedule* of the ∞-searcher for P is simply a polygonal path within P over which he moves. A point $x \in P$ is said to be *contaminated* at time t during the execution of a schedule, if it is possible for an intruder, by some motion over time, to be at x at t without being illuminated at any time t' such that $t' \leq t$. A point that is not contaminated is said to be *clear*. A region $Q \subseteq P$ is clear if every point in Q is clear; otherwise, it is contaminated. A schedule for P is called a *search schedule* if P is clear at the end of the execution. P is said to be *k-searchable* (or *∞-searchable*) if there exists a search schedule of the k-searcher (or ∞-searcher) for P. For formal definitions of these concepts, see [11]. (We do not include an action of "moving while rotating the flashlights in some arbitrary manner," since any such action can be broken down into smaller parts, each of which can be "simulated" using some sequence of the elementary actions. We omit the details.)

As was observed in [11], one of the reasons for the difficulty of deciding whether a given polygon P is k-searchable or ∞-searchable is that some vertices and edges of P may have to be recontai[illegible]ted repeatedly during the search. This observation suggests us to consider a restricted [illegible] of the polygon search problem in which some vertices or edges of P must remain cle[illegible] during the search. Specifically, the problem we consider in this paper is the following.

> Given P and two edges u and v of P, clear P under the following condition $\mathcal{B}$: (1) u must remain clear throughout the search, and (2) at any time, any point on v that is not illuminated is considered to be contaminated.

A possible interpretation of this requirement is that (1) P represents a corridor with two open exits u and v, (2) any number of intruders can leave and enter P through v at any time, and (3) the searcher must force all the intruders out of P through v without allowing any of them to reach u. For t[illegible] version of the problem we present a necessary condition for P to be ∞-searchable, [illegible] show that the same condition is also sufficient for P to be 2-searchable. Therefore, as far as this variant is concerned, the 2-searcher and the ∞-searcher have the same capabi[illegible].

One application of the result presented in this paper is the problem of searching a "multi-level art gallery" consisting of a number of levels of floors (simple polygons) in which every pair of adjacent floors are connected by a staircase whose entrances are located on the respective polygon boundaries. Note that the top and bottom floors have only one entrance, and the intermediate floors have two entrances. One can easily show that to search such a structure, the searcher must proceed from the bottom to the top (or from the top to the bottom) clearing one floor at a time, in such a way that the intruders will not move to the lower floors that have already been searched. This means that every intermediate floor must be cleared in such a way that (1) the entrance to the staircase to the lower floors remains clear, and (2) the intruders are forced out to the upper floors through the entrance to the other staircase. Since the floors are simple polygons and the entrances are their edges, searching any intermediate floor of a multi-level art gallery is exactly the open edge polygon search problem considered in this paper. (Searching the bottom and top floors having only one entrance seems to be much harder than searching the intermediate floors having two entrances. Formally, the one-entrance problem is obtained from the open edge problem stated above by dropping the condition that edge u must remain clear during the search.)

In the rest of this paper, unless otherwise stated, P is an n-sided simple polygon, $u = \overline{u_L u_R}$ is the edge of P that should remain clear during the search, and and $v = \overline{v_L v_R}$ is the edge of P whose points are assumed to be contaminated whenever it is not illuminated. We assume that u_L, v_L, v_R and u_R appear in this order clockwise in the boundary ∂P of P.

2 A necessary condition

Two points x and $y \in P$ are said to be mutually *visible* if $\overline{xy} \subseteq P$. We let $V(x)$ denote the set of points in P that are visible from a point x, and define $V^2(x) = \bigcup_{y \in V(x)} V(y)$. Note that $V(x) \subseteq V^2(x)$. If $y \in V^2(x)$, then we say that y is *link-2-visible* from x. For regions Q and $R \subseteq P$, we say that Q is *weakly visible* (or *weakly link-2-visible*) from R if every point

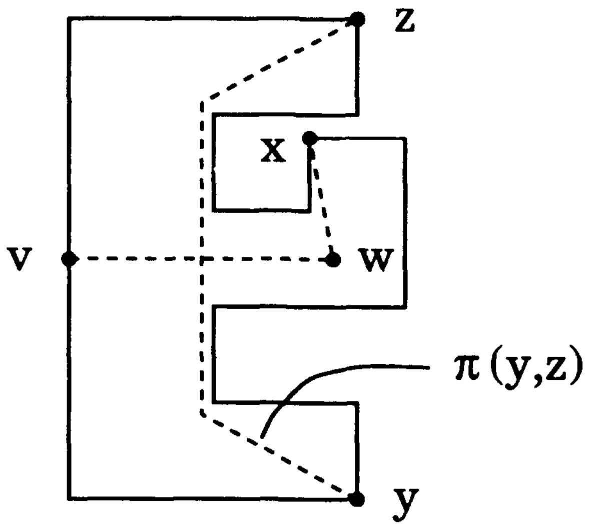

Figure 1: w is visible from x; v is link-2-visible from x; y and z are separable from w and link-2-separable from x.

in Q is visible (or link-2-visible) from some point in R. (In this definition, if R consists of a single point p, then we simply say that Q is visible (or link-2-visible) from p.) For points x, y and $z \in P$, y and z are said to be ***separable*** (or ***link-2-separable***) from x if every path within P between y and z contains at least one point in $V(x)$ (or $V^2(x)$). Note that since P is simple, y and z are separable (or link-2-separable) from x iff $\pi(y,z)$ contains at least one point in $V(x)$ (or $V^2(x)$), where $\pi(y,z)$ is the Euclidean shortest path within P between y and z. See Figure 1 for illustration.

For points x and $y \in \partial P$, we let $\partial P_L(x,y)$ denote the portion of ∂P from x to y taken clockwise (i.e., the "left" boundary of P from x to y). Similarly, we let $\partial P_R(x,y)$ denote the portion of ∂P from x to y taken counterclockwise (i.e., the "right" boundary of P from x to y). $\partial P_L(u_L,v_L)$ and $\partial P_R(u_R,v_R)$ are simply written as ∂P_L and ∂P_R, respectively. For convenience, we use "$<$" to denote the order in which the points in $\partial P_L(u_L,v_L)$ (or $\partial P_R(u_R,v_R)$) appear in a traversal from u_L to v_L (or from u_R to v_R).

For vertices $x \in \partial P_L$ and $y \in \partial P_R$, we say that x and y are in *conflict* with respect to u if (1) u_L and x are not link-2-separable from y and (2) u_R and y are not link-2-separable from x. Similarly, x and y are said to be in *conflict* with respect to v if (1) v_L and x are not link-2-separable from y and (2) v_R and y are not link-2-separable from x. See Figure 2. We say that u (or v) is *conflict-free* if there do not exist such vertices that are in conflict with respect to it. Finally, we say that u and v *satisfy the weak link-2-visibility condition* if ∂P_L is weakly link-2-visible from $\pi(u_R,v_R)$ and ∂P_R is weakly link-2-visible from $\pi(u_L,v_L)$.

Theorem 1 *If P is ∞-searchable under condition B, then (1) u is conflict-free, (2) v is conflict-free, and (3) u and v satisfy the weak link-2-visibility condition.*

Proof (1) Suppose that $x \in \partial P_L$ and $y \in \partial P_R$ are in conflict with respect to u. Then since u_R is illuminated at time zero and $x \notin V^2(u_R)$, x is contaminated at time zero. Similarly, y is contaminated at time zero. If x is illuminated before y at time t, then y and u_R are not separable from the position of the ∞-searcher at t, and hence u_R becomes contaminated. Similarly, if y is illuminated before x, then u_L becomes contaminated. Thus P cannot be

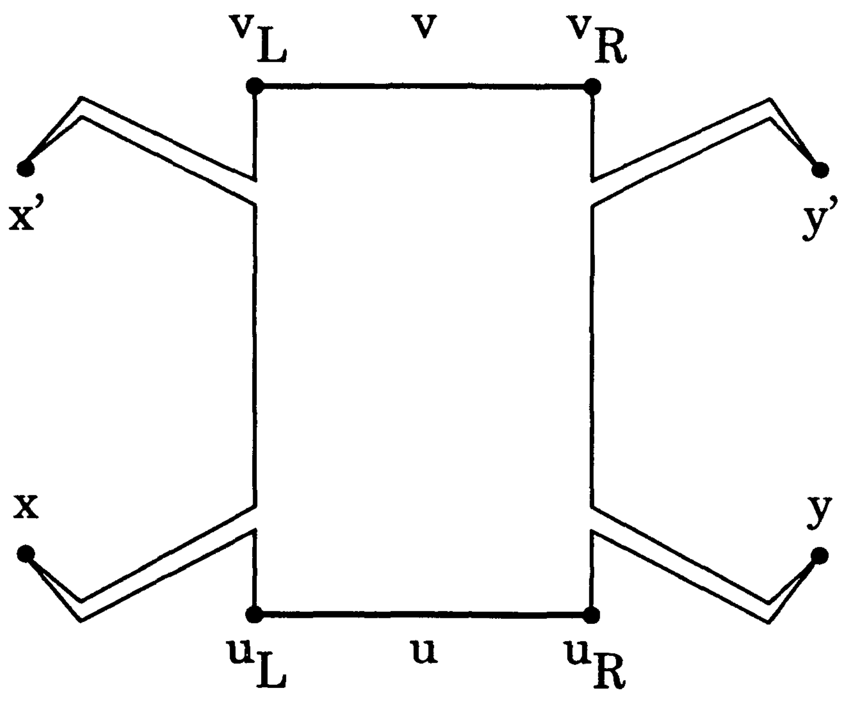

Figure 2: x and y are in conflict with respect to u; x' and y' are in conflict with respect to v.

cleared without contaminating u. **(2)** Suppose that $x \in \partial P_L$ and $y \in \partial P_R$ are in conflict with respect to v. When x is illuminated, y is contaminated since (a) v_R is not illuminated (and hence is contaminated by assumption) and (b) v_R and y are not separable from the location of the ∞-searcher. Similarly, x is contaminated when y is illuminated. Therefore x and y cannot become clear simultaneously. **(3)** Suppose that $x \in \partial P_L$ is not link-2-visible from $\pi(u_R, v_R)$. Then u_R becomes contaminated when x is illuminated, since (a) v_R is not illuminated (and hence is contaminated by assumption) and (b) v_R and u_R are not separable from the location of the ∞-searcher. Similarly, u_L becomes contaminated when $y \in \partial P_R$ not link-2-visible from $\pi(u_L, v_L)$ is illuminated. □

3 Sufficiency

The following theorem, together with Theorem 1 and an obvious fact that any 2-searchable polygon is k-searchable for any $k \geq 3$ and ∞-searchable, shows that the condition given in Theorem 1 is in fact necessary and sufficient for P to be searchable under condition $\mathcal{B}$ by the k-searcher for any $k \geq 2$ and the ∞-searcher.

Theorem 2 *If (1) u is conflict-free, (2) v is conflict-free, and (3) u and v satisfy the weak link-2-visibility condition, then P is 2-searchable under condition $\mathcal{B}$.*

In the rest of this section, we prove Theorem 2. Assume that the conditions of the theorem are satisfied.

Let us denote the two flashlights by F_L (the "left" flashlight) and F_R (the "right" flashlight). Suppose that the 2-searcher is located at point $s \in P$, aiming F_L and F_R at

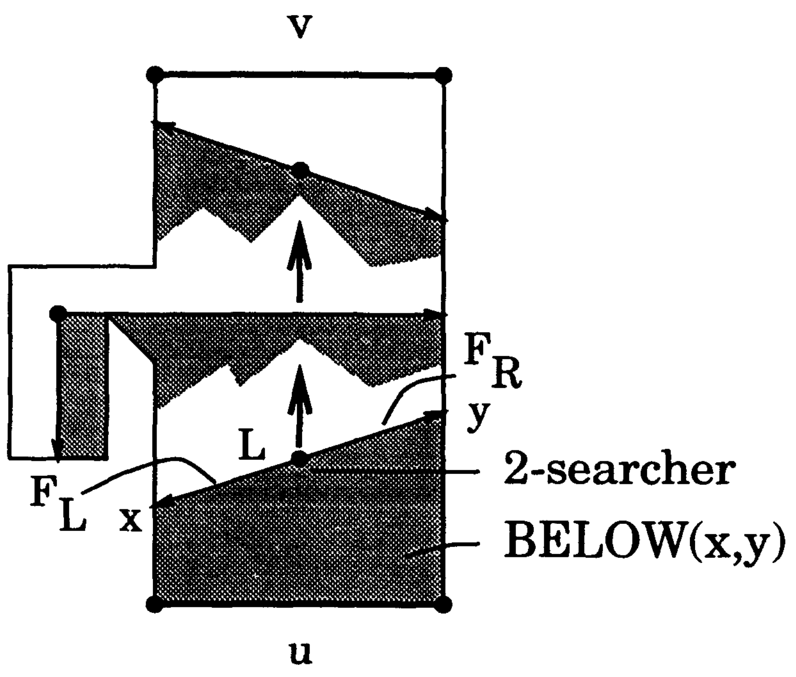

Figure 3: Sweeping P by $\mathcal{L}$ at $\overline{xy}$ keeping BELOW(x, y) clear.

points $x \in \partial P_L$ and $y \in \partial P_R$, respectively. Intuitively, we view the polygonal chain $\overline{xsy}$ as a variable-length two-link chain $\mathcal{L}$ determined by the rays of the flashlights, and clear P by sweeping it by $\mathcal{L}$ from u to v (i.e., $\mathcal{L}$ coincides with edges u and v at the beginning and end, respectively), in such a way that at any time, the subregion of P "below" $\mathcal{L}$ remains clear. While we do so, we keep $\mathcal{L}$ straight as much as possible, bending it at the position of the 2-searcher only when we clear the regions not visible from the opposite boundary. See Figure 3. In the following, for points $x \in \partial P_L$ and $y \in \partial P_R$ that are mutually visible, we denote by BELOW(x,y) the subregion of P "below" $\overline{xy}$, i.e., the region determined by $\overline{xy}$ and that portion of ∂P from x to y taken counterclockwise. Note that BELOW(x,y) contains edge u that must remain clear.

We need the following definitions found in [7]. A *chord* of P is a line segment within P whose endpoints are both in ∂P. For a convex vertex x of P that is adjacent to a reflex vertex x', let c be the chord given as the segment between x' and the first hit point in ∂P of the ray emanating from x through x'. We say that c is *induced* by x. (Each such x can induce up to two chords.) Chord c divides P into two subpolygons, and the one containing x is called the *component* of c and denoted by $P(c)$. Let D be the set of chords induced by such vertices x. Chord $c \in D$ is said to be *redundant* if there exists another chord $c' \in D$ such that $P(c') \subseteq P(c)$. Let $C \subseteq D$ be the set of nonredundant chords. The set C provides useful information for generating a search schedule, since obviously no searcher can clear P unless he visits $P(c)$ for every $c \in C$. It is known [7] that C can be constructed in $O(n \log n)$ time for an n-sided polygon P, using the bullet shooting algorithm of [5] and a modified version of an algorithm given in [10]. (*Bullet shooting* is the problem of finding the first point in ∂P hit by the ray emanating from the given point in P in the given direction.)

Now, we denote by C_L (or C_R) the set of chords $c \in C$ such that both endpoints of c are in ∂P_L (or ∂P_R). It is easy to see that if $\mathcal{L}$ intersects every chord $c \in C_L \cup C_R$ during

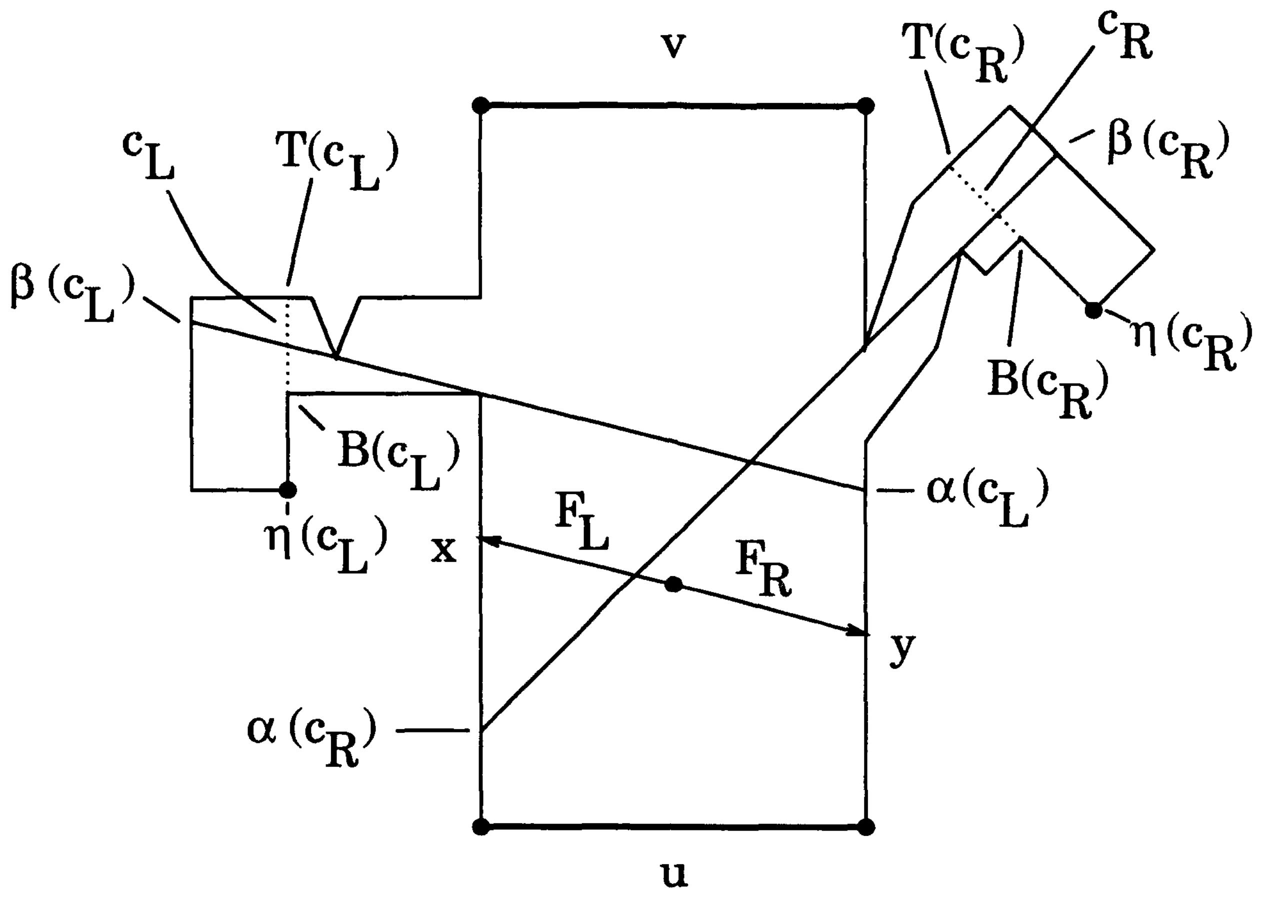

Figure 4: c_L, $\alpha(c_L)$, $\beta(c_L)$, c_R, $\alpha(c_R)$ and $\beta(c_R)$.

the sweep, then every point in P will become visible from the 2-searcher at least once and hence, can be illuminated by one of the flashlights. In fact, we sweep P by repeatedly finding a suitable "next" chord $c \in C_L \cup C_R$ and advancing $\mathcal{L}$ so that it intersects c.

For each chord $c \in C_L \cup C_R$, let $\eta(c)$, $B(c)$ and $T(c)$ be the convex vertex inducing c and the "bottom" and "top" endpoints of c, respectively, where $B(c) < \eta(c) < T(c)$. (See Figure 4.) Suppose that currently, $\mathcal{L}$ is at $\overline{xy}$ for some $x \in \partial P_L$ and $y \in \partial P_R$. First, we find the chord $c_L \in C_L$ whose B point is encountered first in a "forward" traversal of $\partial P_L(x, v_L)$ from x to v_L. (If such c_L does not exist, then we treat vertex v_L as a chord such that $B(v_L) = T(v_L) = v_L$. A similar comment applies to c_R mentioned next.) We find $c_R \in C_R$ in a similar manner. Clearly, either c_L or c_R must be the next chord to be intersected by $\mathcal{L}$. To decide which of the two chords should be intersected next, we do the following. For c_L, let $\alpha(c_L)$ be the first point, encountered in a "forward" traversal of $\partial P_R(y, v_R)$ from y to v_R, from which at least one point in c_L is visible, if such a point exists; otherwise, let $\alpha(c_L)$ be the first point, encountered in a "backward" traversal of $\partial P_R(u_R, y)$ from y to u_R, from which at least one point in c_L is visible. (Such $\alpha(c_L)$ always exists, since u and v satisfy the weak link-2-visibility condition.) Then let p be the point in c_L closest to $B(c_L)$ visible from $\alpha(c_L)$, and let $\beta(c_L) \in \partial P_L$ be the first point at which the ray emanating from $\alpha(c_L)$ through p penetrates ∂P. (Note that $\alpha(c_L)$ and $\beta(c_L)$ are defined relative to x and y.) Again, see Figure 4 for illustration. Note that $\mathcal{L}$ will coincide $\overline{\beta(c_L)\alpha(c_L)}$ if we advance it

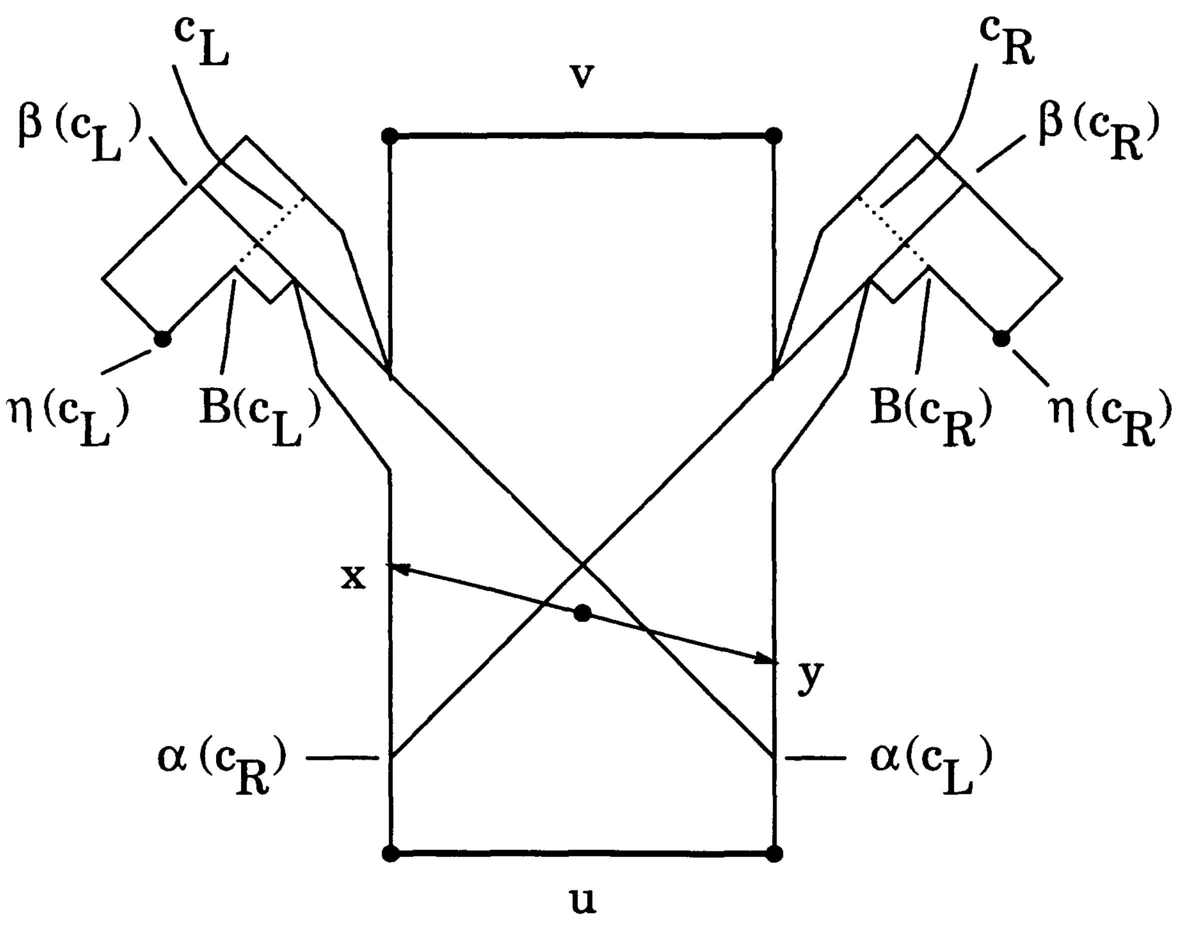

Figure 5: $\eta(c_L)$ and $\eta(c_R)$ are in conflict with respect to v.

without bending, with "minimum" movement, until it intersects c_L. (We say "minimum," since for $\mathcal{L}$ to intersect c_L, the right endpoint of $\mathcal{L}$ must move from y to $\alpha(c_L)$ *at least*.) We also define $\alpha(c_R) \in \partial P_L$ and $\beta(c_R) \in \partial P_R$ for c_R in a completely symmetrical manner.

Lemma 1 *The conditions $\alpha(c_L) < y$ and $\alpha(c_R) < x$ cannot hold simultaneously.*

Proof If $\alpha(c_L) < y$ and $\alpha(c_R) < x$, then $\eta(c_L)$ and $\eta(c_R)$ must be in conflict with respect to v (see Figure 5). This contradicts the assumption. □

So in the following, assume that at least one of $y \leq \alpha(c_L)$ and $x \leq \alpha(c_R)$ holds. There are two cases.

Case 1: $y \leq \alpha(c_L)$ and $x \leq \alpha(c_R)$.
In this case, we have the following lemma.

Lemma 2 *At least one of $\alpha(c_R) \leq B(c_L)$ and $\alpha(c_L) \leq B(c_R)$ holds.*

Proof If $B(c_L) < \alpha(c_R)$ and $B(c_R) < \alpha(c_L)$, then $\eta(c_L)$ and $\eta(c_R)$ must be in conflict with respect to u (see Figure 6). This contradicts the assumption. □

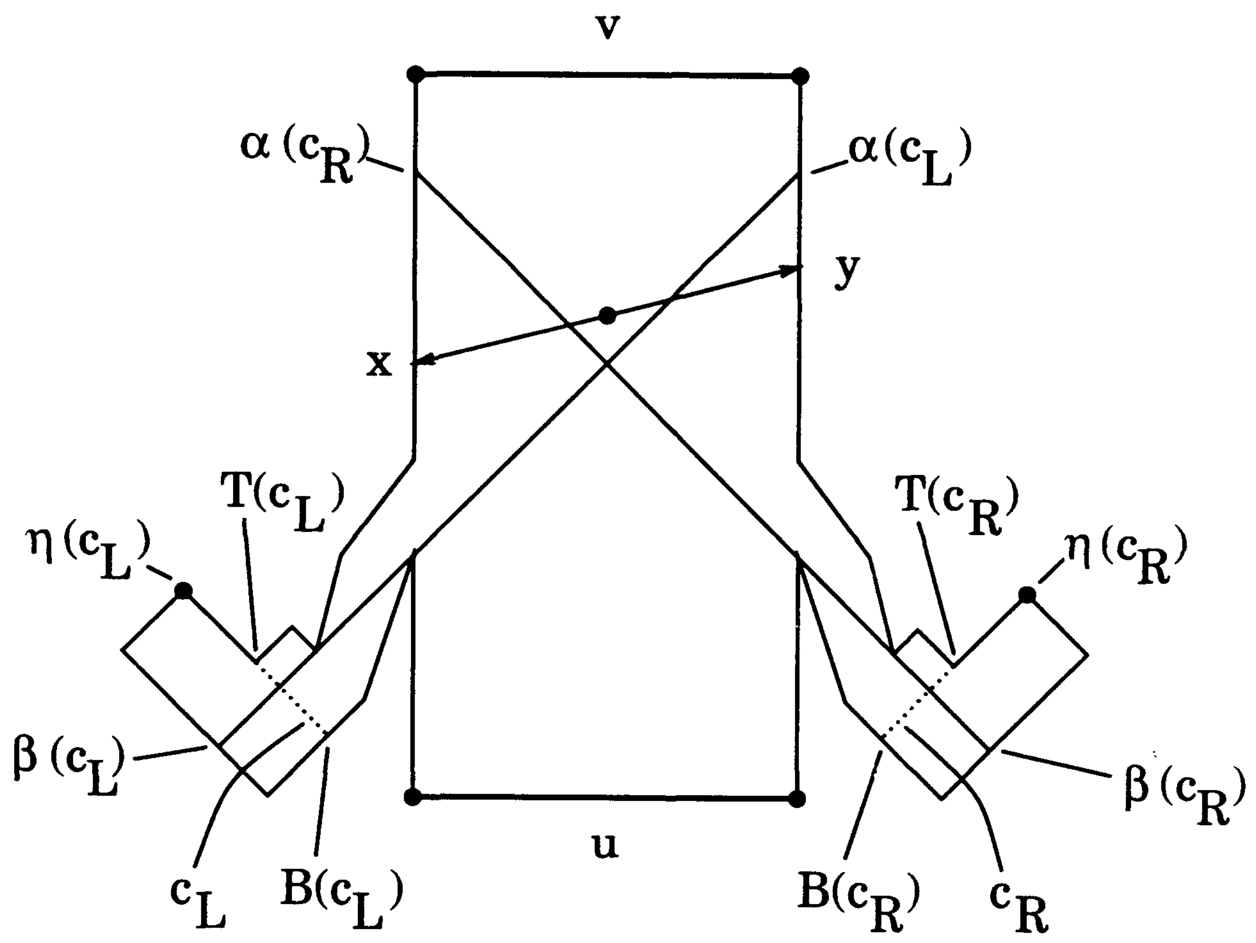

Figure 6: $\eta(c_L)$ and $\eta(c_R)$ are in conflict with respect to u.

```
Algorithm 𝒜:
begin
  Compute $\pi(y, \beta(c_L)) = \overline{s_1 s_2 \ldots s_m}$ where $s_1 = y$, $s_m = \beta(c_L)$, none of $s_1, \ldots, s_{m-1}$ is
  in $\partial P_L(\beta(c_L), v_L)$, and no three consecutive points in $s_1, \ldots, s_m$ are collinear;
  for $i := 1$ to $m - 1$ do
    if $s_i \in \partial P_R$ and $s_{i+1} \in \partial P_R$ then
      begin
        $x_i :=$ SHOOT($s_i, s_{i+1}$);
        L_ADVANCE_BY_SWEEP($x_i$);
        R_ADVANCE_FROM_LID($s_{i+1}$); {$\mathcal{L}$ is at $\overline{x_i s_{i+1}}$.}
      end
    else if $s_i \in \partial P_R$ and $s_{i+1} \in \partial P_L$ then L_ADVANCE_BY_SWEEP($s_{i+1}$); {$\mathcal{L}$ is at $\overline{s_{i+1} s_i}$.}
    else if $s_i \in \partial P_L$ and $s_{i+1} \in \partial P_R$ then R_ADVANCE_BY_SWEEP($s_{i+1}$); {$\mathcal{L}$ is at $\overline{s_i s_{i+1}}$.}
    else {$s_i \in \partial P_L$ and $s_{i+1} \in \partial P_L$}
      begin
        $x_i :=$ SHOOT($s_{i+1}, s_i$); {$x_{m-1} =$ SHOOT($s_m, s_{m-1}$) $= \alpha(c_L)$}
        R_ADVANCE_BY_SWEEP($x_i$);
        L_ADVANCE_FROM_LID($s_{i+1}$); {$\mathcal{L}$ is at $\overline{s_{i+1} x_i}$.}
      end;
end;
```

Figure 7: Algorithm $\mathcal{A}$ for advancing $\mathcal{L}$ from $\overline{xy}$ to $\overline{\beta(c_L)\alpha(c_L)}$.

If $\alpha(c_L) \leq B(c_R)$, then we advance $\mathcal{L}$ to $\overline{\beta(c_L)\alpha(c_L)}$ so that it intersects c_L. (Otherwise, $\alpha(c_R) \leq B(c_L)$ holds by Lemma 2, and we advance $\mathcal{L}$ to $\overline{\alpha(c_R)\beta(c_R)}$ so that it intersects c_R. This is a symmetric case, and thus we omit the details.) This is done by algorithm $\mathcal{A}$ shown in Figure 7. Algorithm $\mathcal{A}$ is written using the following four operations and SHOOT, where SHOOT(r, s) is the first point at which the ray emanating from point r through point s intersects ∂P. In the following explanations, assume that $\mathcal{L}$ is currently at $\overline{xy}$ and BELOW(x, y) is clear.

1. L_ADVANCE_FROM_LID(z): z is a point in $\partial P_L(x, v_L)$ such that (1) z, x and y are collinear and (2) $\partial P_L(x, z)$ is weakly visible from $\overline{xz}$. See Figure 8. The 2-searcher moves to x aiming F_L and F_R at x and y, respectively, and then clears the region whose boundary is $\partial P_L(x, z) \cup \overline{xz}$ by sweeping $\partial P_L(x, z)$ using F_L, while moving along the "lid" $\overline{xz}$ of the region and aiming F_R continuously at y. This is possible since $\partial P_L(x, z)$ is weakly visible from $\overline{xz}$. When this is done, $\mathcal{L}$ is at $\overline{zy}$ and BELOW(z, y) is clear.

2. R_ADVANCE_FROM_LID(z): This is symmetric to L_ADVANCE_FROM_LID(z), and can be used to advance $\mathcal{L}$ to $\overline{xz}$.

3. L_ADVANCE_BY_SWEEP(z): z is a point in $\partial P_L(x, v_L)$ such that (1) $z \in V(y)$ and (2) $\partial P_L(x, z)$ is link-2-visible from y. See Figure 9. The 2-searcher moves to y aiming F_L and F_R at x and y, respectively, and then sweeps $\partial P_L(x, z)$ from "pivot"

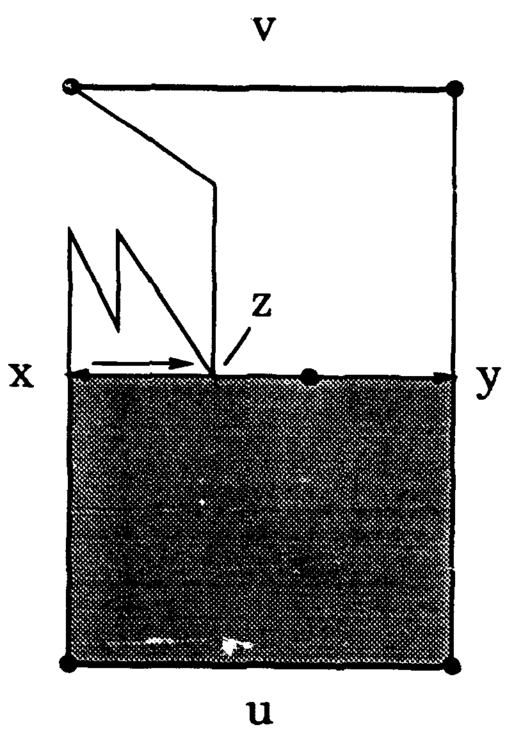

Figure 8: Illustration for L_ADVANCE_FROM_LID(z).

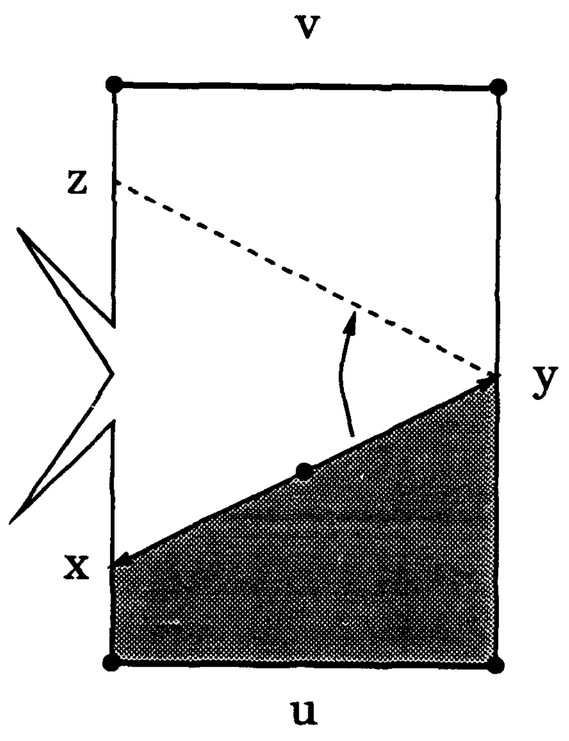

Figure 9: Illustration for L_ADVANCE_BY_SWEEP(z).

y using F_L, in such a way that each time F_L is advanced to the "lid" of a portion of $\partial P_L(x,z)$ not visible from y, he, while aiming F_R continuously at y, (1) moves to the lid aiming F_L at the lid, (2) sweeps the portion by F_L using the method given in L_ADVANCE_FROM_LID, and then (3) returns to y aiming F_L at the lid. This is possible since $\partial P_L(x,z)$ is link-2-visible from y. When this is done, $\mathcal{L}$ is at $\overline{zy}$ and BELOW(z,y) is clear.

4. R_ADVANCE_BY_SWEEP(z): This is symmetric to L_ADVANCE_BY_SWEEP(z), and can be used to advance $\mathcal{L}$ to $\overline{xz}$.

The process of advancing $\mathcal{L}$ by algorithm $\mathcal{A}$ is illustrated in Figure 10.

Lemma 3 *The four operations used in algorithm $\mathcal{A}$ can always be executed successfully.*

Proof Since u and v satisfy the weak link-2-visibility condition, any region cleared by L_ADVANCE_FROM_LID and R_ADVANCE_FROM_LID is weakly visible from its "lid." So these two operations can always be executed successfully. For L_ADVANCE_BY_SWEEP and R_ADVANCE_BY_SWEEP, we need to show that any region cleared by either of them is weakly link-2-visible from the "pivot" of the sweep (e.g., point y of Figure 9). But this indeed is the case, since by the way c_L is selected, (1) there are no chords in C_L between x and $B(c_L)$, and (2) there are no chords in C_R between y and $\alpha(c_L)$. □

Case 2: $\alpha(c_L) < y$ or $\alpha(c_R) < x$.
Suppose that $\alpha(c_L) < y$ holds. (The argument for the case $\alpha(c_R) < x$ is similar, and is thus omitted.) Then we advance $\mathcal{L}$ from $\overline{xy}$ to $\overline{\beta(c_L)\alpha(c_L)}$ as follows. See Figure 11 for illustration. Let q be the intersection of $\overline{xy}$ and $\overline{\beta(c_L)\alpha(c_L)}$. Here, $\partial P_L(x,\beta(c_L))$ is link-2-visible from q, since otherwise, there must exist a chord in C_L between x and $B(c_L)$, contradicting the way c_L is selected. Thus we move the 2-searcher to q aiming F_L and F_R at x and y, respectively, and then sweep $\partial P_L(x,\beta(c_L))$ from q using F_L by a method similar to L_ADVANCE_BY_SWEEP with pivot q, with a slight modification that (1) F_L and F_R are aimed in opposite directions whenever the 2-searcher is located at q, and (2) F_R is aimed through q whenever the 2-searcher is not located at q. Since F_R is rotated only clockwise, BELOW$(\beta(c_L),\alpha(c_L))$ becomes clear when the sweep is completed. Note that the right endpoint of $\mathcal{L}$ has been moved backward from y to $\alpha(c_L)$. Thus we refer to this case as a *back-up* case.

Note that the endpoints of $\mathcal{L}$ are moved backward toward u only when a back-up case occurs, and a back-up case can cause some chords to be selected more than once as the next chord to which $\mathcal{L}$ is advanced. However, we have the following lemma.

Lemma 4 *A chord to which $\mathcal{L}$ is advanced and that generates a back-up case can never be selected again as the next chord to which $\mathcal{L}$ is advanced.*

Proof Suppose that a chord, say $c_L \in C_L$, generated a back-up case when $\mathcal{L}$ was advanced to it. Assume that c_L is selected again later as the next chord to which $\mathcal{L}$ is advanced. Then there must have been another back-up case that brought the left endpoint of $\mathcal{L}$ to a point below $B(c_L)$. If that back-up case was caused by a chord $c_R \in C_R$ such that $\alpha(c_L) < B(c_R)$, then $\eta(c_L)$ and $\eta(c_R)$ must be in conflict with respect to v (see Figure 5). Otherwise, by an elementary analysis, we can show that there must exist chords $c'_L \in C_L$ and $c_R \in C_R$ such that $\eta(c'_L)$ and $\eta(c_R)$ are in conflict with respect to v (see Figure 12). In either case, the assumption that v is conflict-free is violated. □

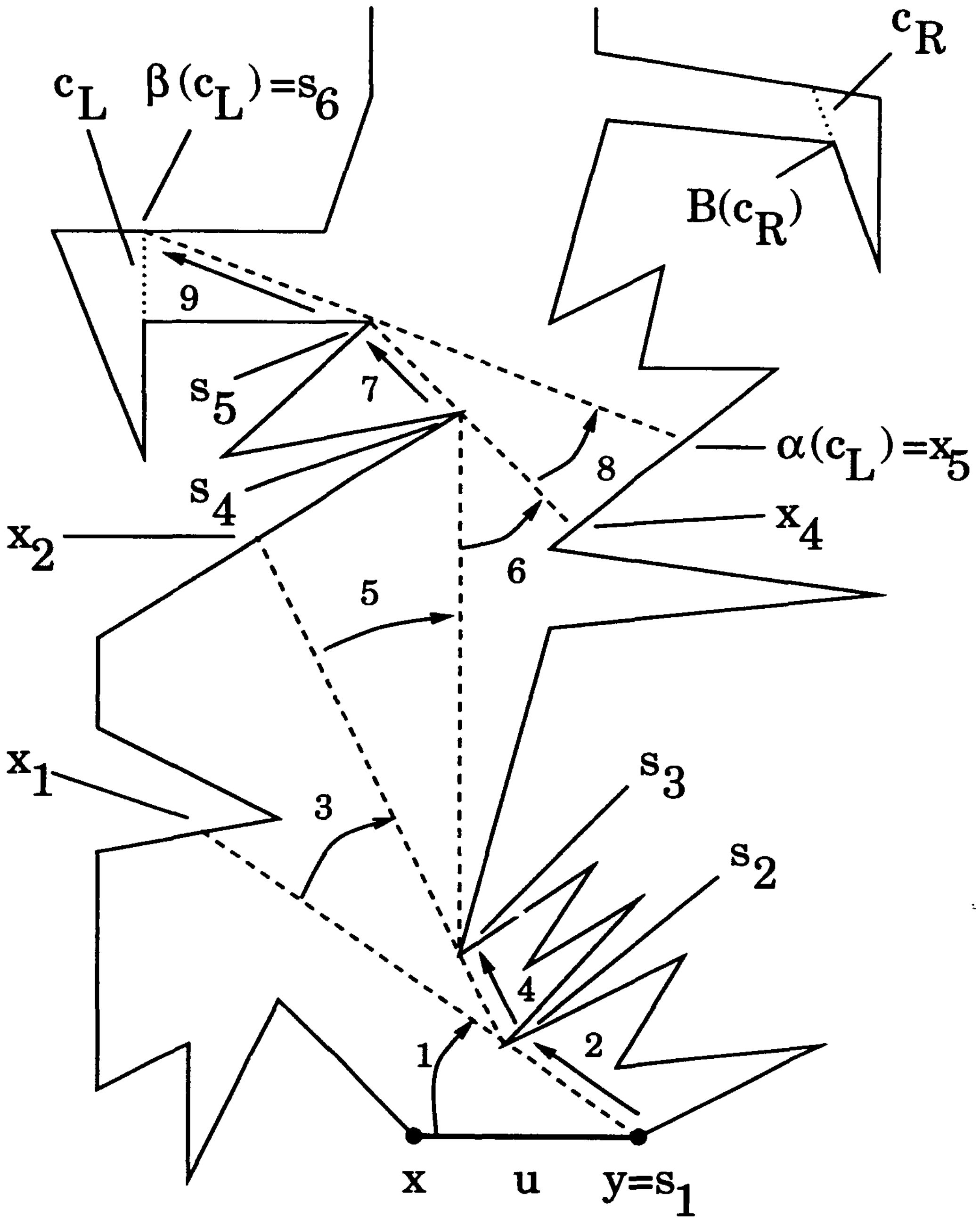

Figure 10: Advancement of $\mathcal{L}$ from $\overline{xy} = u$ to $\overline{\beta(c_L)\alpha(c_L)}$ by algorithm $\mathcal{A}$ of Figure 7. 1, 3, 5: L_ADVANCE_BY_SWEEP; 2, 4: R_ADVANCE_FROM_LID; 6, 8: R_ADVANCE_BY_SWEEP; 7, 9: L_ADVANCE_FROM_LID.

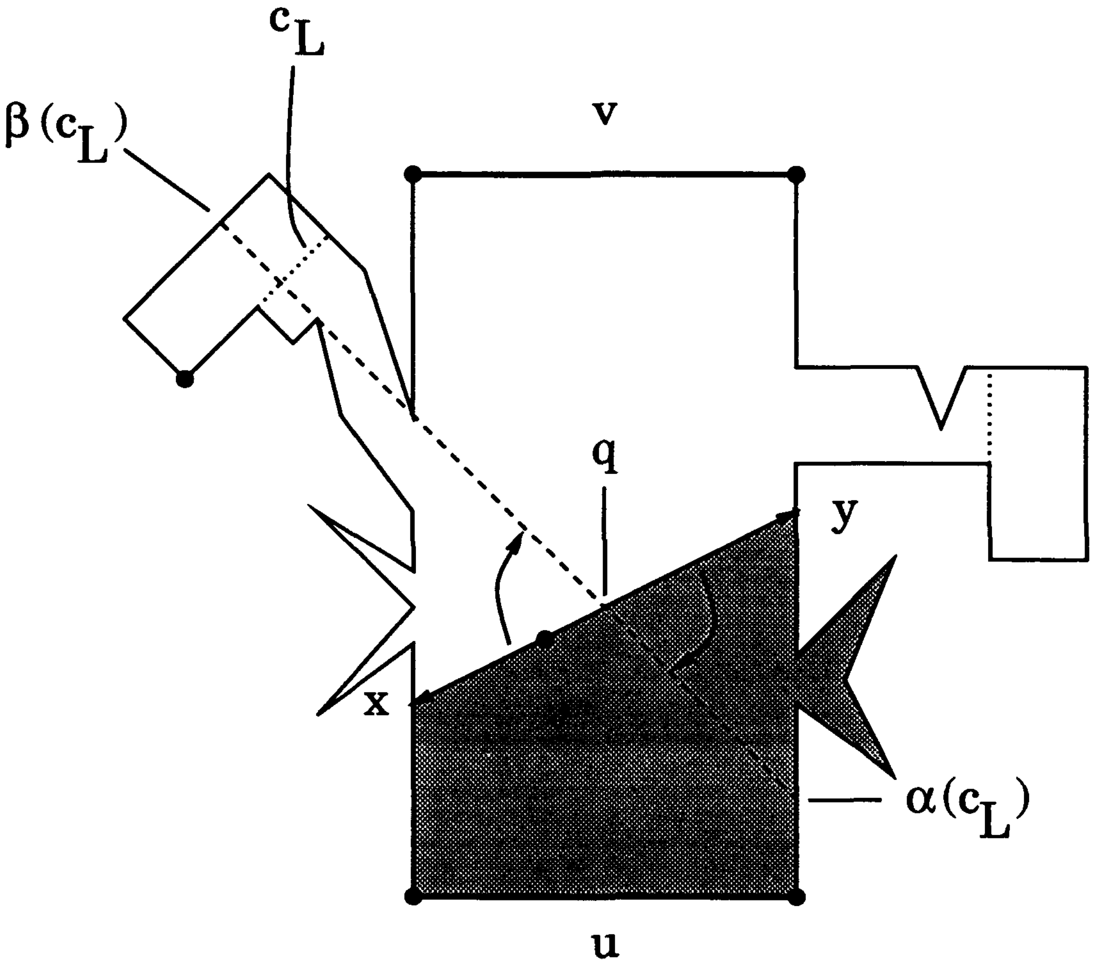

Figure 11: Advancement of $\mathcal{L}$ from $\overline{xy}$ to $\overline{\beta(c_L)\alpha(c_L)}$ in a back-up case.

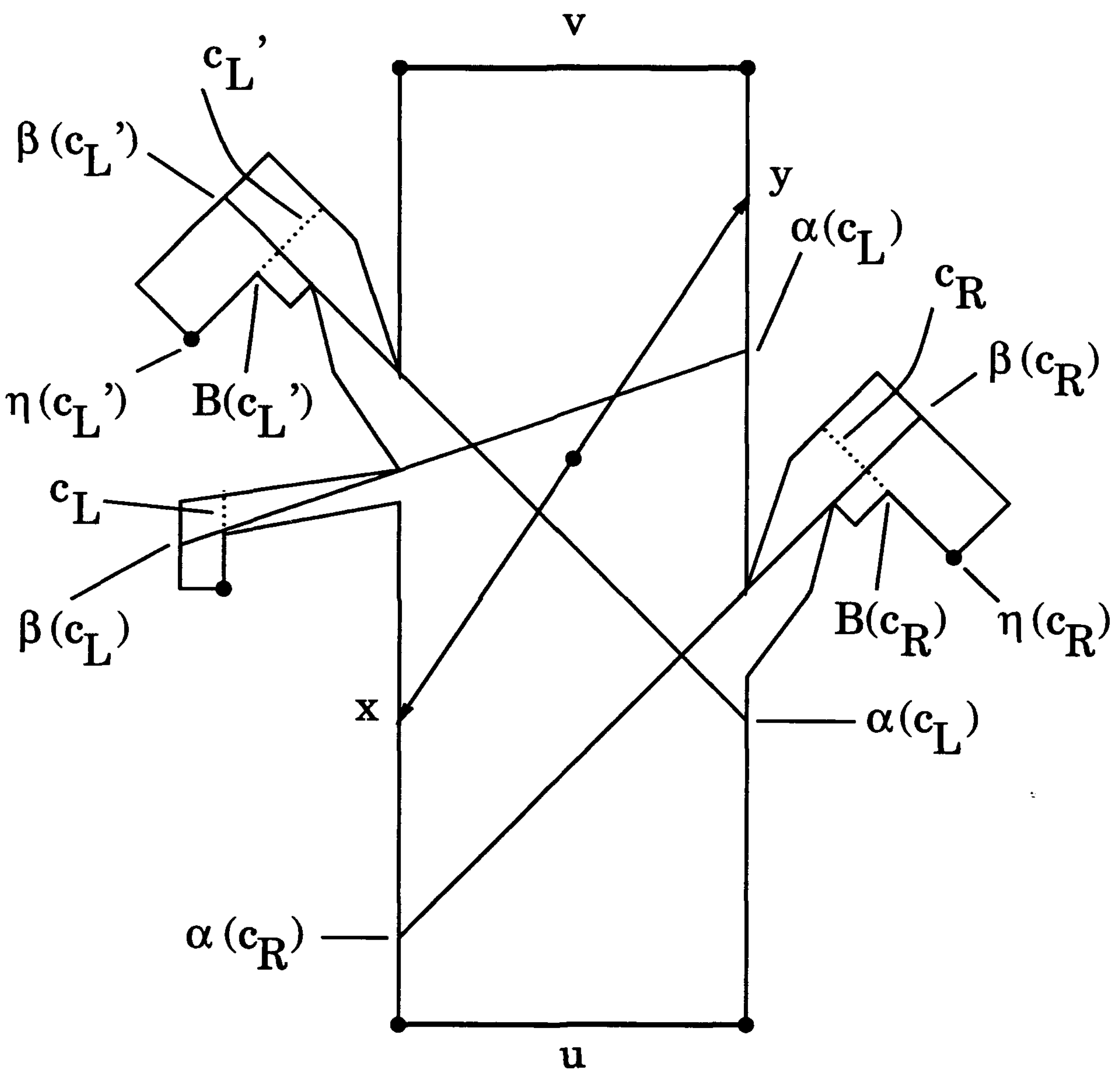

Figure 12: $\eta(c'_L)$ and $\eta(c_R)$ are in conflict with respect to v.

By Lemma 4, there can be only $O(n)$ back-up cases during the entire sweep, and hence each chord in $C_L \cup C_R$ can be selected as the next chord to which $\mathcal{L}$ is advanced only $O(n)$ times. (End of Case 2)

Therefore, eventually $\mathcal{L}$ is advanced to $v = \overline{v_L v_R}$, and P becomes clear. This completes the argument for proving Theorem 2.

4 Time complexity

For arbitrary vertices x, y and z of P, y and z are not link-2-separable from x iff y and z belong to the same maximal connected region of $P - V^2(x)$. This condition can be tested for any given vertices x, y and z in constant time, once we construct $V^2(x)$ for e[illegible] vertex x and find, for each vertex $y \notin V^2(x)$, the maximal connected region of $P - V^2($[illegible]aining y. Since $V^2(x)$ can be constructed in $O(n)$ time for each vertex x from a triang[illegible]on of P [10], where n is the number of vertices of P, whether there exist vertices that are in conflict with respect to u or v can be tested in $O(n^2)$ time. Since whether u and v satisfy the weak link-2-visibility condition can also be tested in $O(n^2)$ time by constructing $V^2(x)$ for each vertex x, we can test whether P satisfies the condition of Theorem 2 in $O(n^2)$ time.

Theorem 3 *If P satisfies the condition of Theorem 2, then a search schedule of th[illegible] 2-searcher consisting of $O(n^2)$ elementary actions for clearing P under condition $\mathcal{B}$ can be generated in $O(n^2 \log n)$ time.*

Proof First, we construct the sets C_L and C_R in $O(n \log n)$ time [7] [10]. Then we sort, in $O(n \log n)$ time, the B ("bottom") endpoints of the chords in C_L (or C_R) in the order they appear in ∂P_L (or ∂P_R). The sorted lists of the B endpoints allow us to determine, each time $\mathcal{L}$ is advanced to a new location, the next chord (forward of $\mathcal{L}$) on each side of P in $O(\log n)$ time. Next, we add the T and B endpoints of the chords in $C_L \cup C_R$ to the vertex set of P (if they are not vertices), and compute a triangulation of P using the extended vertex set. This can be done in $O(n)$ time [1]. From this triangulation, we construct, in linear time, a data structure given in [5] that solves the bullet shooting problem in $O(\log n)$ time per query. Also, for each vertex of the extended vertex set of P, we construct, in linear time, a data structure given in [5] that allows us to compute the Euclidean shortest path from the vertex to an arbitrary point in P in $O(\log n + m)$ time, where m is the number of segments in the path. The total time needed for the preprocessing of P described above is $O(n \log n)$.

Let us first discuss the case in which no back-up case occurs. Let M be the number of times that $\mathcal{L}$ is advanced to the next chord, where $M = O(n)$. Consider the i-th time when $\mathcal{L}$ is advanced, where currently $\mathcal{L}$ is at $\overline{xy}$. We first find the next chords $c_L \in C_L$ and $c_R \in C_R$ forward of $\mathcal{L}$ in $O(\log n)$ time. Then we compute two shortest paths $\pi(y, B(c_L))$ and $\pi(y, T(c_L))$ from the data structure we constructed, and obtain the points $\alpha(c_L)$ and $\beta(c_L)$ by (1) examining where the two shortest paths separate from each other and (2) using the $O(\log n)$ time bullet shooting algorithm. We obtain the points $\alpha(c_R)$ and $\beta(c_R)$ in a similar manner, and then determine the next chord to which $\mathcal{L}$ is advanced in additional constant time.

When the next chord, say c_L, is determined, we use algorithm $\mathcal{A}$ to generate a schedule of the 2-searcher. Note that the shortest path $\pi(y, \beta(c_L))$ that we use in algorithm $\mathcal{A}$ has in effect been computed when we obtained $\beta(c_L)$. Clearly the length of the schedule is linear in the number m of vertices between $\overline{xy}$ and $\overline{\beta(c_L)\alpha(c_L)}$, and the schedule itself can be generated in $O(m \log n)$ time by (1) bullet-shooting from some of these vertices, (2) triangulating each of the regions swept by L_ADVANCE_BY_SWEEP and R_ADVANCE_BY_SWEEP (to find the subregions not visible from the pivot of the sweep in additional linear time), and (3) processing each of the vertices in constant time a constant number of times. Since there are no back-up cases, during the M times $\mathcal{L}$ is advanced, each vertex of P appears only a constant number of times in the shortest paths that are constructed and in the M schedules that are generated. Thus the total time needed for constructing the shortest paths is $O(M \log n + n) = O(n \log n)$, and the additional time needed for generating the M schedules is $O(n \log n)$. Therefore the total time needed to generate the entire schedule, whose length is obviously $O(n)$, is $O(n \log n)$ (including the $O(n \log n)$ time preprocessing).

Now assume that back-up cases can occur. Clearly the length of the schedule in a back-up case for a chord, say $c_L \in C_L$, is linear in the number of vertices between x and $\beta(c_L)$, and the schedule itself can be generated by processing each such vertex in constant time a constant number of times after triangulating the region to be swept and obtaining the subregions not visible from the pivot of the sweep. Also, by using an argument similar to that given above, we can show that the length of the schedule *between* consecutive back-up cases and the time needed to generate it are $O(n)$ and $O(n \log n)$, respectively. Therefore, since there can be only $O(n)$ back-up cases (as we discussed in the previous section), the length of the entire schedule and the time needed to generate it (including the preprocessing) are $O(n^2)$ and $O(n^2 \log n)$, respectively. □

The length of the schedule generated in the proof of Theorem 2 is asymptotically worst-case optimal, as is shown in Example 1.

Example 1 Consider the n-sided polygon P shown schematically in Figure 13, where $m \in \Theta(n)$ is even. We show that any search schedule of the ∞-searcher for P under condition $\mathcal{B}$ contains $\Omega(n^2)$ elementary actions. Note that (1) for each $2 \le i \le m$, $V(c_1), \ldots, V(c_{i-1})$ must be clear when the searcher visits $V(c_i)$, and (2) since $V(d_1), \ldots, V(d_m)$ become contaminated when the searcher visits $V(c_i)$ for even i, the searcher must first visit each of $V(d_1), \ldots, V(d_m)$ each time he visits $V(c_j)$ for odd j. Thus each of $V(d_1), \ldots, V(d_m)$ must be visited $m/2$ times before $V(c_1), \ldots, V(c_m)$ become clear simultaneously. Since $m \in \Theta(n)$, this implies that any search schedule must contain $\Omega(n^2)$ elementary actions. □

5 Concluding Remarks

We have considered the open edge variant of the polygon search problem. We have shown that the 2-searcher has the same capability as the ∞-searcher in this problem, and presented a simple necessary and sufficient condition for the existence of a search schedule. We have also presented an algorithm for generating a search schedule and discussed its time complexity.

As a final note, we remark that the "two guards problem" considered in [6] in which one moves two guards from vertices u to v of a polygon along the two boundary chains determined by u and v in such a way that they always remain mutually visible, is closely related to searching the given polygon using a single 1-searcher (having one flashlight) under the assumption given in this paper. Clearly, if there exists a schedule for the two

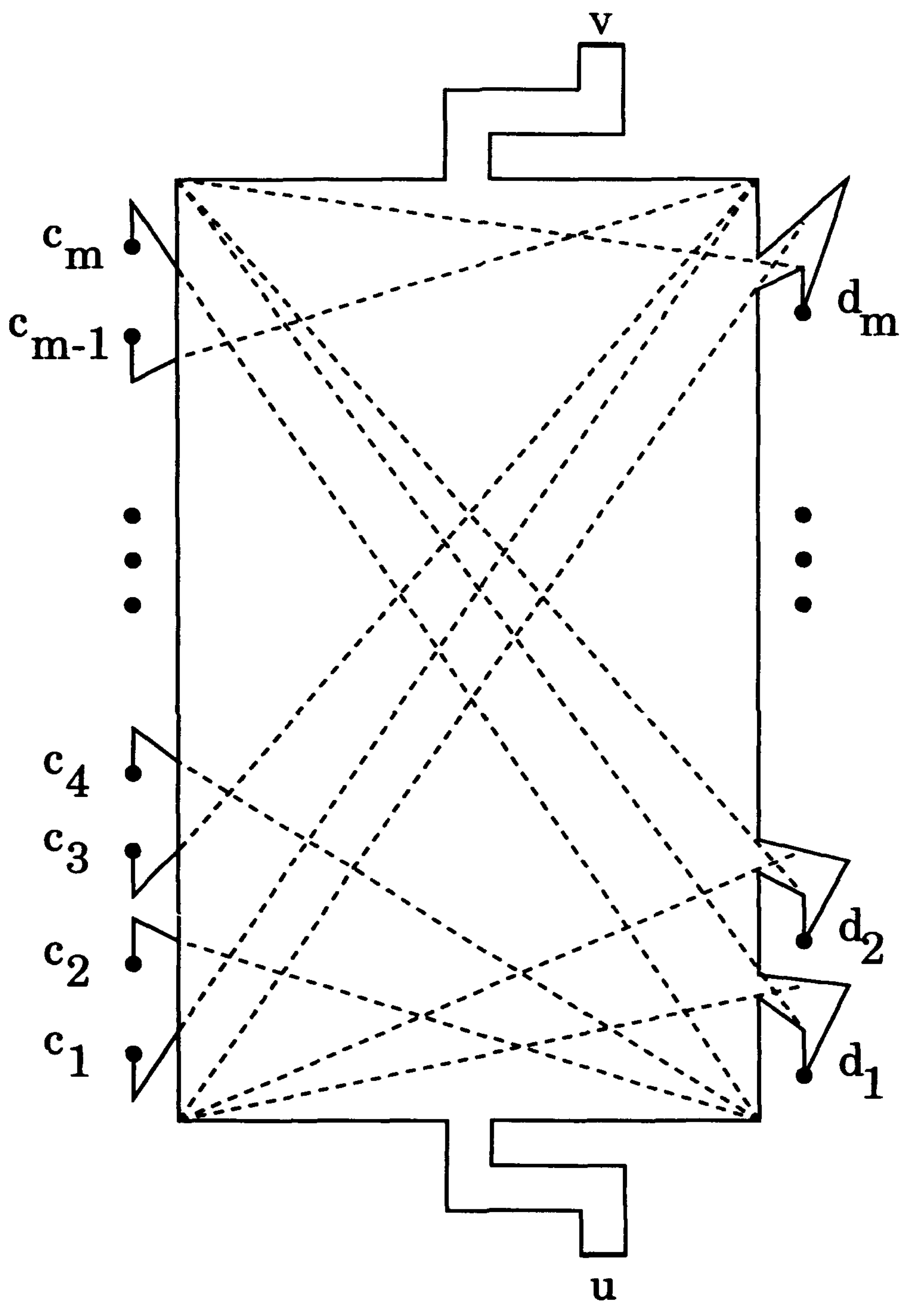

Figure 13: An n-sided polygon requiring a schedule of length $\Omega(n^2)$.

guards, then the 1-searcher can search the polygon by behaving as one guard and aiming the flashlight at the position of the other guard. It is not clear whether the converse is also true, since the beam of the flashlight can move backwards "jumping over a dent," whereas the two guards must always move "smoothly" over the polygon boundary. However, we conjecture that any schedule for the 1-searcher can be converted to that for two guards, since it seems unlikely that a polygon that requires such backward moves can actually be searched by the 1-searcher using a different schedule. Studying the capability of the 1-searcher, as well as the challenging problem of generating a schedule for any searcher in the one-entrance case mentioned in Section 1, are suggested for future research.

Acknowledgment We wish to thank the anonymous referees for their helpful comments on an earlier version of this paper.

References

[1] B. Chazelle, "Triangulating a simple polygon in linear time," *Proc. of the 31st Symposium on Foundations of Computer Science*, St. Louis, Missouri, October 1990, pp. 220–230.

[2] W.-P. Chin and S. Ntafos, "Optimum watchman routes," *Proc. of the 2nd Annual Symposium on Computational Geometry*, Yorktown Heights, New York, June 1986, pp. 24–33.

[3] W.-P. Chin and S. Ntafos, "Shortest watchman routes in simple polygons," Technical Report, Computer Science Dept., Univ. of Texas at Dallas, 1987.

[4] V. Chvátal, "A combinatorial theorem in plane geometry," *J. Combinatorial Theory (B) 18*, 1975, pp. 39-41.

[5] L. Guibas, J. Hershberger, D. Leven, M. Sharir and R. Tarjan, "Linear-time algorithms for visibility and shortest path problems inside triangulated simple polygons," *Algorithmica 2*, 1987, pp. 209–233.

[6] C. Icking and R. Klein, "The two guards problem," *Proc. of the 7nd Annual Symposium on Computational Geometry*, North Conway, New Hampshire, June 1991, pp. 166–175.

[7] Y. Ke, "Polygon visibility algorithms for weak visibility and link distance problems," Ph.D. Dissertation, Department of Computer Science, Johns Hopkins University, 1989.

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

[9] K. Sugihara, I. Suzuki and M. Yamashita, "The searchlight scheduling problem," *SIAM J. Computing 19*, No. 6, 1990, pp. 1024–1040.

[10] S. Suri, "Minimum link paths in polygons and related problems," Ph.D. Dissertation, Department of Computer Science, Johns Hopkins University, 1987.

[11] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM J. Computing 21*, No. 5, 1992, pp. 863–888.