International Journal of Computational Geometry & Applications

THE TWO GUARDS PROBLEM

CHRISTIAN ICKING

and

ROLF KLEIN

Praktische Informatik VI, Fernuniversität – GH – Hagen
Elberfelder Str. 95, D-5800 Hagen, Germany

Received 18 June 1991
Revised 18 December 1991
Communicated by D. T. Lee

ABSTRACT

Given a simple polygon in the plane with two distinguished vertices, s and g, is it possible for two guards to simultaneously walk along the two boundary chains from s to g in such a way that they are always mutually visible? We decide this question in time $O(n \log n)$ and in linear space, where n is the number of edges of the polygon. Moreover, we compute a walk of minimum length within time $O(n \log n + k)$, where k is the size of the output, and we prove that this is optimal.

Keywords: visibility, watchman routes, planar polygons, motion planning.

1. Introduction

Several types of watchman problems (also called art gallery problems) have been proposed in the literature. The first of this family of problems was the *minimum watchmen* problem considered by Chvatal[1] and Fisk[2] and many others, where a number, as small as possible, of places for watchmen in a given scene is to be calculated such that every point of the edges (walls) in the scene is visible from at least one place (watchman). Lee and Lin[3] have shown that finding the exact minimum number of watchmen in a simple polygon is NP-hard.

Another example of this type of problem is the *watchman route* problem. There, a route of minimum length within a polygon has to be found such that every point of the polygon is visible from at least one point of the route. In some circumstances, this problem is NP-hard, too, see Chin and Ntafos[4]. A survey of art gallery problems and many other related issues can be found in O'Rourke[5].

In this paper, we study the following problem. Imagine a police patrol in a dangerous midtown street. Each of two officers has to walk along one of the sidewalks, all the way down the street, and to check all the bank doors on his side. Can the two guards proceed in such a way that they are always mutually visible?

More formally, let us assume that we are given a simple polygon with n edges and distinguished vertices s and g. Can two points be moved from s to g, each on one of the two boundary chains, such that the line segment connecting the two points is at each time fully contained in the polygon? The points are allowed to backtrack locally, but eventually they must both arrive at g. A movement subject to these constraints is called a *walk*. A polygon that admits a walk is *walkable*. The *general walk problem* asks for a walk in which the total distance travelled by the two points is minimized.

One special case arises when no backtracking is necessary in order to get the points from s to g. Such a walk will be called *straight*. During a straight walk, the line segment connecting the points *sweeps* the polygon in an ordered way. In a sense, this is a generalization of *monotone polygons*, which can be swept with a line of fixed orientation. In our problem, the line can turn during the sweep. As we compute a walk of minimum length, we will find a straight walk, if there is one. It turns out that the important information for deciding if a straight walk exists can be extracted from the hit points of the edge extensions of the polygon's reflex vertices, see Figure 6. For answering these shooting queries in total time $O(n \log n)$, we employ the method of Chazelle and Guibas[6]. In Section 3 we show that this leads to an $O(n \log n)$ algorithm that computes a straight walk.

During a walk it may be necessary for one of the guards to move backwards (towards s) while the other one still advances towards g. This type of motion is called a *counter-walk*. In Section 4 we give an $O(n \log n)$ algorithm for the straight counter-walk problem: Can one guard move from s to g while the other one moves from g to s on the other side without backtracking, in such a way that they are always mutually visible? In addition to answering shooting queries, here we also have to compute certain visibilities. To this end, we apply the shortest path algorithm by Guibas and Hershberger[7] in order to compute, for each vertex v, the first segments of the shortest paths from v to a point on the opposite chain. This can be done in total time $O(n \log n)$.

Section 5 shows how these results can be used to solve the general walk problem, i.e. to move the guards from s to g with backtracking. We compute in time $O(n \log n + k)$ a minimum length walk, and prove in Section 6 that this is the optimum bound. Here, k denotes the size of the output, i. e. the number of walk instructions for the two guards. A single walk instruction causes the guards to walk straight towards the next halting point, at most to the next vertex in the given direction. Hence, the number k of walk instructions is not less than the total number of vertex visits, but of the same order of magnitude. As we will show in Section 6, k can be of order $\Theta(n^2)$.

2. Notations

We are given a *simple polygon* P in the plane, i.e. a polygon without self-intersections or holes. Two vertices s and g are marked; then the polygon's boundary consists of two polygonal chains, L and R, with common endpoints s and g. Points on L will be denoted by p, p_1, p' or similarly, points on R by q, q_1, q', etc.

Succ(p) (resp. Pred(p)) denote the vertex next (resp. previous) to p in a chain. Both chains, L and R, are oriented from s to g, see Figure 1.

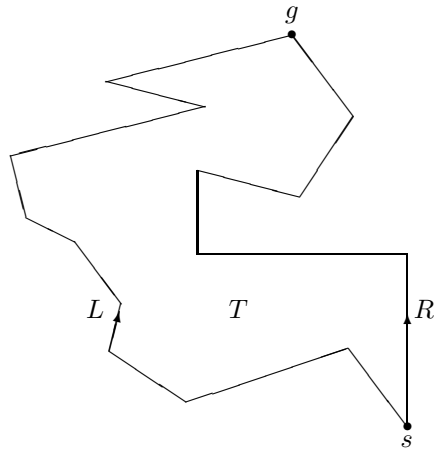


Fig. 1. Polygon P consisting of two chains L and R

Just for convenience, we assume that the polygon is in a general position in the following sense. No three lines extending three edges of P have a point in common, and no three vertices are collinear.

Definition 2.1 Let T denote the union of the interior domain of polygon P and P itself. We call a point $x \in P$ *visible* from a point $y \in P$ iff the connecting line segment $\overline{xy}$ is entirely contained in T. For the parts of P visible from a point x, see Figure 2, we use

$$\mathrm{vis}(x) := \{y \in P \mid y \text{ visible from } x\}$$

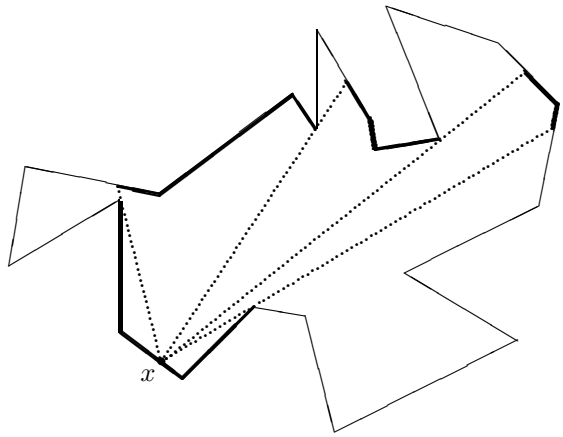


Fig. 2. The points of P visible from x, vis(x)

Visibility is a symmetric relation. We often refer to $\overline{pq}$ as the *visibility segment* of the pair (p, q). Obviously, two visibility segments $\overline{p_1q_1}$ and $\overline{p_2q_2}$ *cross* iff the order of p_1, p_2 on L is opposite to the order of q_1, q_2 on R, see Figure 3.

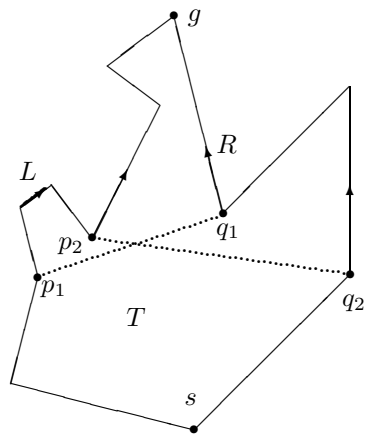


Fig. 3. Crossing visibility segments

Definition 2.2

(i) A *walk* on P is a pair (l, r) of continuous functions such that:

$l : [0,1] \longrightarrow L, \qquad r : [0,1] \longrightarrow R,$

$l(0) = r(0) = s, \qquad l(1) = r(1) = g,$

$l(t)$ is visible from $r(t)$ for all $t \in [0,1]$.

Any line segment $\overline{l(t)\ r(t)}$ is called a *walk line segment* of the walk. The point $r(t)$ is the *walk partner* of $l(t)$, and vice versa.

(ii) A walk on P is called *straight* if both l and r are non-decreasing with respect to the orientation of L and R.

(iii) P is called *(straight) walkable* if it admits a (straight) walk.

(iv) Correspondingly, a *straight walk on P from line segment $\overline{p_0\ q_0}$ to line segment $\overline{p_1\ q_1}$*, where $p_0 < p_1$, $q_0 < q_1$, and $\overline{p_0\ q_0}$, $\overline{p_1\ q_1}$ are contained in P, has to fulfill the conditions $l(0) = p_0$, $r(0) = q_0$, $l(1) = p_1$, $r(1) = q_1$; and l and r are non-decreasing.

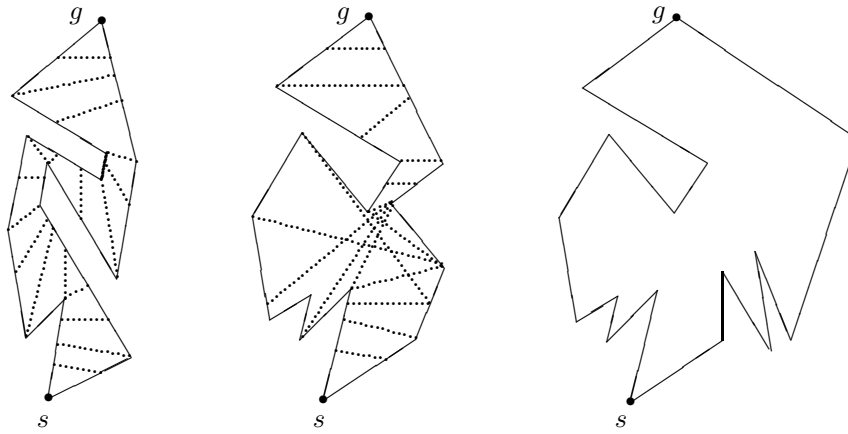


Fig. 4. A straight walkable and a walkable polygon, and one that is not walkable.

Lemma 2.3 *A walk is straight iff no two walk line segments cross.*

We want to decide if P is walkable and, if so, to compute a *minimum length* walk, that is, one where the sum of the lengths of the two curves given by the parametrizations l and r becomes a minimum. Any straight walk (if one exists) is of minimum length. Figure 4 depicts a straight walk, a walk on a polygon that admits no straight walk, and a polygon that is not walkable at all.

Definition 2.4 A *walk instruction* is one of the following elementary motions, see Figure 5.

(i) Both guards move forward along segments of single edges.

(ii) One of the guards moves forward, the other one moves backward along segments of single edges.

As a special case of (i) or (ii) one of the guards may stand still while the other moves.

Let k denote the number of walk instructions in a walk of minimal length.

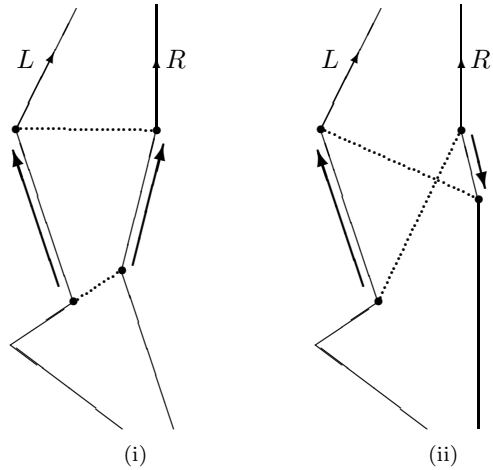


Fig. 5. The types of walk instructions.

3. Straight Walks

Our algorithm for solving the straight walk problem consists of the following steps.

First, we solve a special shooting problem. Let r be a *reflex vertex* of P (i.e. one with interior angle larger than 180°). For each of the two rays emanating from vertex r through the interior of the polygon that extend the incident edges we determine the first hit points on P, see Figure 6. Then we build up a balanced search tree for all vertices and hit points on L, and a second one for the vertices and hit points on R. Links are established between those points on L and R that are origin and target of the same shot.

Based on this data structure, a straight walk can be computed in the following way. First, we check if a situation like the one depicted in Figure 7 (i) occurs. If so, the polygon is clearly not straight walkable. Otherwise, we proceed by computing,

for each vertex v, a connected subset of the polygonal chain, L or R, opposite to v that will be called the *walk interval* of v. It turns out that the polygon is straight walkable if and only if none of the walk intervals is empty. In this case, the walk interval of vertex $p \in L$ consists of all points $q \in R$ such that there exists a straight walk of the polygon in which the walk line segment $\overline{pq}$ occurs (note that it is *not* a priori clear why these sets should be connected!).

We now describe our algorithm in detail. For a point v and a direction $\vec{u}$, let $r(v, \vec{u})$ denote the ray starting from v with the direction of $\vec{u}$. We are interested in the following shooting queries.

Definition 3.1 Let p_{j-1}, p_j, p_{j+1} be three consecutive vertices on L, where p_j is a reflex vertex. We denote by $\mathrm{Forw}(p_j)$ the first point of P on the ray $r(p_j, \overrightarrow{p_{j-1}p_j})$ that is visible from p_j. Analogously, $\mathrm{Backw}(p_j)$ is the first point of $r(p_j, \overrightarrow{p_{j+1}p_j}) \cap P$ that is visible from p_j. For vertices on R, Forw and Backw are defined symmetrically.

See an example in Figure 6. We may consider Forw and Backw as functions that return just the first point of P hit by the corresponding ray.

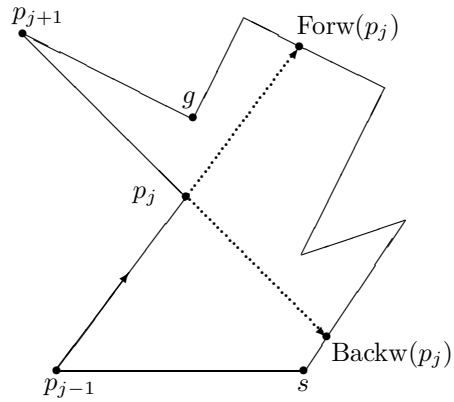


Fig. 6. Forw(p_j) and Backw(p_j)

Chazelle and Guibas[6] have presented an algorithm that computes, after an $O(n \log n)$ time preprocessing step, for any point v in T and for any direction $\vec{u}$ the first intersection of $r(v, \vec{u})$ with P in time $O(\log n)$. Clearly, this result can be used to determine the points Forw(v), Backw(v) for all vertices $v \in L \cup R$ in time $O(n \log n)$, since $O(n)$ shooting queries have to be answered in time $O(\log n)$ each.

For all vertices and hit points on L we build a search structure according to the total order on L, and an analogous structure for R. We establish links between these two search structures, namely for each hit point we establish a link to its corresponding vertex, and for each reflex vertex v two links to Forw(v) and Backw(v).

This structure can be built up incrementally, as the shooting queries are answered, in total time $O(n \log n)$, using balanced trees.

P can only be walkable if L and R are mutually weakly visible. As usual (see Avis and Toussaint[8]), L is called *weakly visible* from R if for each point $p \in L$

there exists a point $q \in R$ such that q is visible from p. Here we give an equivalent characterization for the chain L being weakly visible from R.

Lemma 3.2 *The chain L is weakly visible from R iff there is no reflex vertex $p \in L$ such that*

$$p < \mathrm{Backw}(p) \in L \qquad or \qquad p > \mathrm{Forw}(p) \in L$$

holds.

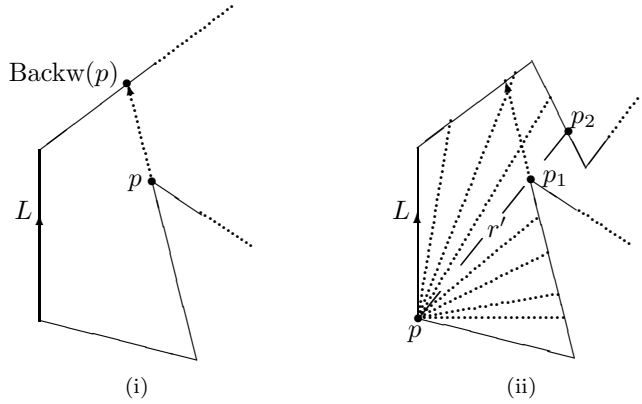


Fig. 7. Weak visibility and hit points (Lemma 3.2)

Proof. Necessity is obvious, since if, for example, $p < \mathrm{Backw}(p) \in L$ then $\mathrm{Succ}(p)$ is invisible from R, see Figure 7 (i).

For sufficiency, if a point $p \in L$ cannot see any point of R, then there must exist a vertex $p_1 \in L$ whose Backw or Forw shot violates the assumed conditions, see Figure 7 (ii). □

The conditions of this lemma can easily be checked on inserting the hit points into the search structure. If any of the conditions becomes true, P is reported not to be straight walkable. In fact, it is not even walkable in that case.

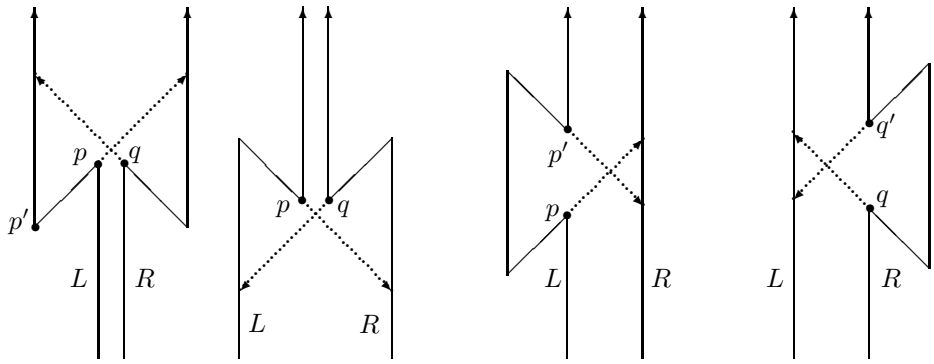
Weak visibility of the chains L and R is a necessary condition for straight walkability. Further necessary conditions are given by the following lemma illustrated by Figure 8. Later in Theorem 3.11 we will see that the conditions listed in Lemma 3.2 and Lemma 3.3 together are also sufficient for straight walkability.

Lemma 3.3 *There is no straight walk for P if one of the following is true, see Figure 8.*

$$\textit{(i)}\ \exists p \in L, q \in R: \quad \Big(q < \mathrm{Backw}(p) \in R \quad \textit{and} \quad p < \mathrm{Backw}(q) \in L\Big) \quad \textit{"deadlock"}$$
$$\textit{or}\Big(q > \mathrm{Forw}(p) \in R \quad \textit{and} \quad p > \mathrm{Forw}(q) \in L\Big)$$
$$\textit{(ii)}\ \exists p, p' \in L: \quad p < p' \quad \textit{and} \quad R \ni \mathrm{Forw}(p') < \mathrm{Backw}(p) \in R \qquad \textit{"left wedge"}$$
$$\textit{(iii)}\ \exists q, q' \in R: \quad q < q' \quad \textit{and} \quad L \ni \mathrm{Forw}(q') < \mathrm{Backw}(q) \in L \qquad \textit{"right wedge"}$$

In case (i) not even a walk for P exists.

Proof. Assume that the first alternative of situation (i) applies and that a walk exists. Let $\widetilde{p}$ be the *first* walk partner of $\mathrm{Succ}(q)$ and $\widetilde{q}$ the first walk partner of



(i) Deadlocks. (ii) Left wedge. (iii) Right wedge.

Fig. 8. Lemma 3.3.

Succ(p). The three points Succ(p), p, and Backw(p) lie on one line. The point q and Backw(q) lie on opposite sides of that line, thus Succ(p) < Backw(q). $L_{<\mathrm{Backw}(q)}$ is not visible from Succ(q). It follows that Succ(p) < Backw(q) $\leq \widetilde{p}$ and, symmetrically, Succ(q) < Backw(p) $\leq \widetilde{q}$.

In consequence, a walk (which is a continuous function) must go through Succ(p) *before* reaching Succ(q) for the first time, and it must pass Succ(q) before reaching Succ(p) for the first time, which is a contradiction, see Figure 8 (i). For obvious reasons, we sometimes call this type of situation a *deadlock*.

In situation (ii), let q and q' denote the walk partners of Succ(p) and Pred(p') in a prospective straight walk. Then $q' \leq \mathrm{Forw}(p') < \mathrm{Backw}(p) \leq q$, contradicting Succ($p$) < Pred($p'$). The other situation can be dealt with analogously, see Figure 8 (ii) and (iii). □

This shows that the Forward and Backward shots of the reflex vertices place restrictions on the possible positions of a walk partner for a given vertex, which in some cases may even cause a polygon to be not straight walkable. As we will see, these constraints are the only ones we need to observe, and if none of the above situations applies, a polygon is in fact straight walkable.

We proceed by computing for each vertex v an interval $[\mathrm{lo}(v), \mathrm{hi}(v)]$ of the opposite chain such that, according to the shot restrictions, any possible walk partner of v must be contained in this interval. For simplicity, in the following definition we assume that p belongs to L. The case of vertices in R is symmetric.

Definition 3.4 For a vertex $p \in L$ we define:

$$\begin{aligned}
\mathrm{hiP}(p) &:= \min\{q \mid q \text{ vertex in } R \text{ and } L \ni \mathrm{Backw}(q) > p\} \\
\mathrm{hiS}(p) &:= \min\{\mathrm{Forw}(p') \in R \mid p' \text{ vertex in } L_{>p}\} \\
\mathrm{hi}(p) &:= \min\{\mathrm{hiP}(p), \mathrm{hiS}(p), g\} \\
\mathrm{loP}(p) &:= \max\{q \mid q \text{ vertex in } R \text{ and } L \ni \mathrm{Forw}(q) < p\} \\
\mathrm{loS}(p) &:= \max\{\mathrm{Backw}(p') \in R \mid p' \text{ vertex in } L_{<p}\} \\
\mathrm{lo}(p) &:= \max\{\mathrm{loP}(p), \mathrm{loS}(p), s\}
\end{aligned}$$

In our notations, the letters S and P stand for *self* and *partner*, correspondingly, meaning that the restriction is caused by a vertex on the same chain or by a vertex on the opposite chain. See Figure 9 for examples how loS, loP, and lo can look like. The following lemmata are direct consequences of the given definition.

Lemma 3.5 *Considered as functions*

$$\{\textit{vertices of } L\} \to R \quad \textit{or} \quad \{\textit{vertices of } R\} \to L,$$

hi *and* lo *are monotonically increasing.*

Lemma 3.6 *Each possible straight walk partner for a vertex v is contained in the interval* $[\mathrm{lo}(v), \mathrm{hi}(v)]$.

Note that, for example, in Figure 8 (i) we have $\mathrm{lo}(p') \geq \mathrm{Backw}(p) > q \geq \mathrm{hi}(p')$, so $[\mathrm{lo}(p'), \mathrm{hi}(p')]$ is empty.

The above quantities can be calculated in two passes over the search structures, one from s to g to calculate loS, loP, and lo, and one from g to s for the others. This takes time $O(n)$.

The functions lo and hi are related by the following formal symmetry.

Lemma 3.7 *Let $p \in L$, $q \in R$ be two vertices of P.*

(i) If $q < \mathrm{lo}(p)$ then $\mathrm{hi}(q) < p$; if $q > \mathrm{hi}(p)$ then $\mathrm{lo}(q) > p$.

(ii) $p \in [\mathrm{lo}(q), \mathrm{hi}(q)] \iff q \in [\mathrm{lo}(p), \mathrm{hi}(p)]$.

Proof.

(i) We consider two cases, namely $\mathrm{lo}(p) = \mathrm{loS}(p)$ and $\mathrm{lo}(p) = \mathrm{loP}(p)$.
Assume $q < \mathrm{lo}(p) = \mathrm{loS}(p)$. According to the definition of loS, there is a vertex $p' \in L_{<p}$ such that $q < \mathrm{lo}(p) = \mathrm{Backw}(p') \in R$. According to the definition of hiP, $\mathrm{hi}(q) \leq \mathrm{hiP}(q) \leq p' < p$.
Assume $q < \mathrm{lo}(p) = \mathrm{loP}(p)$. Denote $\mathrm{loP}(p)$ by q'. This is a vertex in $R_{>q}$ with the property that $L \ni \mathrm{Forw}(q') < p$. Then $\mathrm{hi}(q) \leq \mathrm{hiS}(q) \leq \mathrm{Forw}(q') < p$.
The second assertion of (i) is symmetric to the first.

(ii) If $q \notin [\mathrm{lo}(p), \mathrm{hi}(p)]$ then $q < \mathrm{lo}(p)$ or $q > \mathrm{hi}(p)$. Assume $q < \mathrm{lo}(p)$ then, using (i), $\mathrm{hi}(q) < p$, thus $p \notin [\mathrm{lo}(q), \mathrm{hi}(q)]$.

□

Lemma 3.8 *Let the chains L and R be mutually weakly visible and let $p \in L$ be a vertex satisfying*

$$\mathrm{lo}(p) \leq \mathrm{hi}(p) \ .$$

Then the interval $[\mathrm{lo}(p), \mathrm{hi}(p)]$ *is visible from p.*

Proof. How can p not be visible from $\mathrm{lo}(p)$? The polygon P must somehow intersect $\overline{p\,\mathrm{lo}(p)}$. By definition of lo, there is a point $p' < p$ which is visible from $\mathrm{lo}(p)$, see Figure 9 for the two main cases $\mathrm{lo}(p) = \mathrm{loS}(p)$ and $\mathrm{lo}(p) = \mathrm{loP}(p)$. In one case, $\mathrm{lo}(p) = \mathrm{Backw}(p')$, in the other $p' = \mathrm{Forw}(\mathrm{lo}(p))$. Any intersection of $\overline{p\,\mathrm{lo}(p)}$ from below $\overline{p'\,\mathrm{lo}(p)}$ must intersect $\overline{p'\,\mathrm{lo}(p)}$ first, which is impossible. Thus, $L_{<p'} \cup R_{<\mathrm{lo}(p)}$ does not intersect $\overline{p\,\mathrm{lo}(p)}$, and analogously there is a point $p'' > p$ such that $L_{>p''} \cup R_{>\mathrm{hi}(p)}$ does not intersect $\overline{p\,\mathrm{hi}(p)}$.

Since $\mathrm{lo}(p) \leq \mathrm{hi}(p)$, we conclude that $L_{<p'} \cup L_{>p''} \cup R_{<\mathrm{lo}(p)} \cup R_{>\mathrm{hi}(p)}$ intersects neither $\overline{p\,\mathrm{lo}(p)}$ nor $\overline{p\,\mathrm{hi}(p)}$, see the third drawing in Figure 9. Now assume

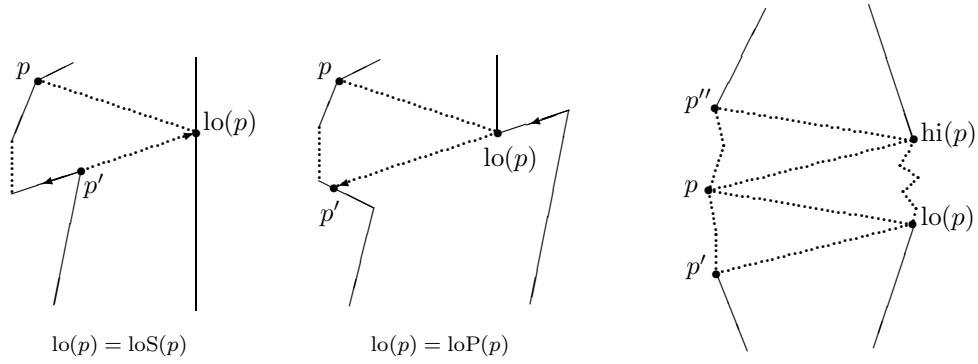


Fig. 9. The cases discussed in Lemma 3.8

that $[p', p)$ intersects $\overline{p\,\mathrm{lo}(p)}$. Then there is a vertex $p''' < p$ such that either $\mathrm{Backw}(p''') \in L_{>p'''}$, see Figure 10 (i), or $\mathrm{Backw}(p''') > \mathrm{lo}(p)$, see Figure 10 (ii); both are contradictions. Analogously, $(p, p'']$ cannot intersect $\overline{p\,\mathrm{hi}(p)}$ or $\overline{p\,\mathrm{lo}(p)}$. Thus, no part of L can intersect $\overline{p\,\mathrm{lo}(p)}$ and $\overline{p\,\mathrm{hi}(p)}$.

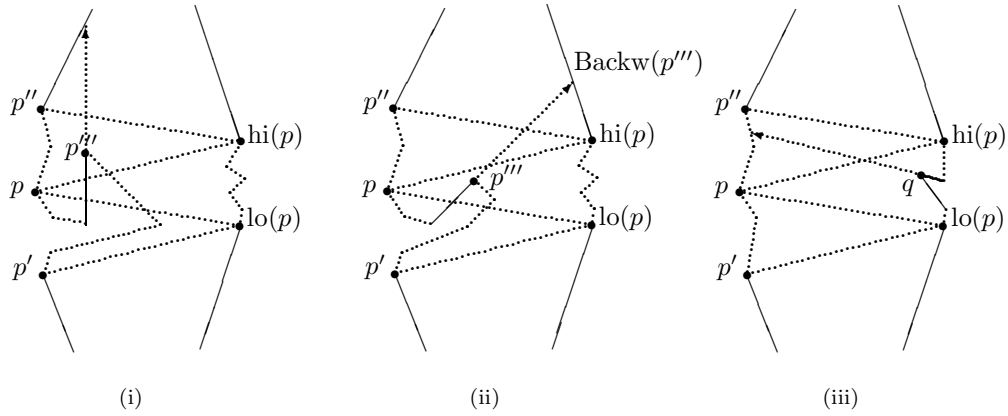


Fig. 10. Cases leading to a contradiction in the proof of Lemma 3.8

Furthermore, by definition of lo and hi, there is no vertex $q \in (\mathrm{lo}(p), \mathrm{hi}(p))$ such that $\mathrm{Forw}(q) < p$ or $\mathrm{Backw}(q) > p$, see Figure 10 (iii) for a case contradicting the definition of hiP. Thus, every point in $[\mathrm{lo}(p), \mathrm{hi}(p)]$ is visible from p. □

The following Lemma 3.9 gives a characterization for a polygon to be straight walkable. The conditions stated there can be checked in $O(n)$ time.

Lemma 3.9 *Let the chains L and R be mutually weakly visible. There is a straight walk for P iff*

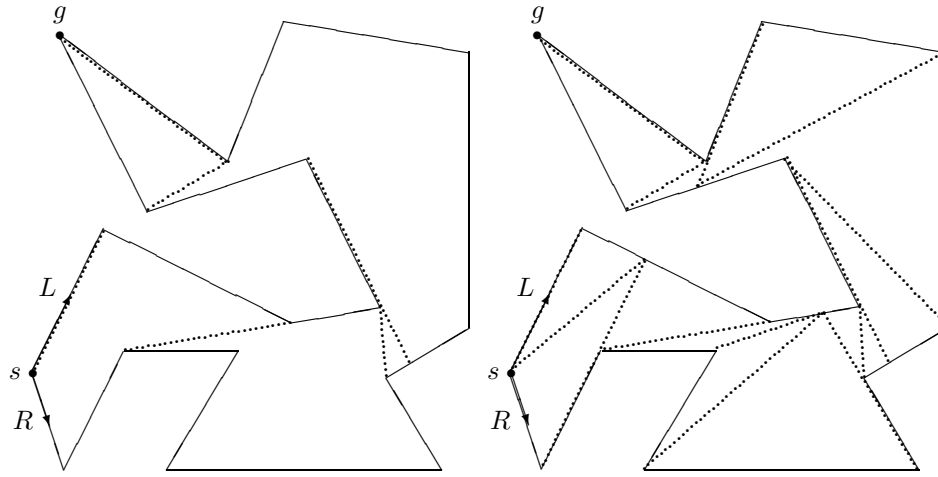
$$\mathrm{lo}(v) \le \mathrm{hi}(v) \text{ for all vertices } v \in L \cup R.$$

Proof. Necessity follows from Lemma 3.6.

In order to prove sufficiency we will explicitly construct a straight walk. In Lemma 3.8 we have seen that the interval $[\mathrm{lo}(v), \mathrm{hi}(v)]$ is visible from v, for all vertices v. The monotonicity property, Lemma 3.5, of lo guarantees that no two

walk line segments cross if we choose (and mark in the data structure) $\mathrm{lo}(p)$ as a walk partner for each vertex $p \in L$.

A vertex $q \in R$ that has not yet been assigned a walk partner lies between two nearest marked points, $\mathrm{lo}(p)$ and $\mathrm{lo}(p')$, where $p < p'$. It follows from Lemma 3.7 that $p < \mathrm{hi}(q) < p'$. Since function hi is monotonic, we can choose $\mathrm{hi}(q)$ as a walk partner for q, and all these walk line segments do not cross, see Figure 11 (ii). The polygon is now subdivided into a sequence of quadrilaterals and triangles, which gives us a sequence of walk instructions. □



(i) All vertices of L have a walk partner (ii) All vertices have a walk partner

Fig. 11. Construction of a straight walk

The choice of $\mathrm{lo}(p)$ as a walk partner for p and $\mathrm{hi}(q)$ for q is quite arbitrary. For symmetry, we may also choose $\mathrm{hi}(p)$ as a walk partner for all vertices $p \in L$. This has the following consequence.

Corollary 3.10 *If P is straight walkable then for every vertex v each point of the interval $[\mathrm{lo}(v), \mathrm{hi}(v)]$ is a suitable walk partner for v.*

Proof. There is one walk including $\overline{p\,\mathrm{lo}(p)}$ as a walk line segment, and another one including $\overline{p\,\mathrm{hi}(p)}$. Now we consider the following walk. From s to $\overline{p\,\mathrm{lo}(p)}$ we follow the first walk, then we rotate the walk line segment around p until it is in the position of $\overline{p\,\mathrm{hi}(p)}$, and continue with the second walk towards g. The rotation is possible because the whole interval $[\mathrm{lo}(p), \mathrm{hi}(p)]$ is visible from p, according to Lemma 3.8. □

Because of this property we call the interval $[\mathrm{lo}(p), \mathrm{hi}(p)]$ the *walk interval* of p.

Now it is not difficult to prove the sufficiency of the conditions in Lemma 3.2 and Lemma 3.3.

Theorem 3.11 *Let the chains L and R be mutually weakly visible. Then there is a straight walk for P iff none of the conditions of Lemma 3.3 is true. To test the conditions, and to construct a straight walk, $O(n \log n)$ time and linear space are sufficient.*

Proof. To apply Lemma 3.9, we have to show that $\mathrm{lo}(v) \leq \mathrm{hi}(v)$ for all vertices $v \in L \cup R$. Assume that $\mathrm{lo}(p) > \mathrm{hi}(p)$ for some $p \in L$. Then $\mathrm{lo}(p) \neq s$ and $\mathrm{hi}(p) \neq g$. Four cases have to be checked: $\mathrm{hi}(p)$ is either $\mathrm{hiP}(p)$ or $\mathrm{hiS}(p)$, $\mathrm{lo}(p)$ is either $\mathrm{loS}(p)$ or $\mathrm{loP}(p)$.

Case 1: $\mathrm{hi}(p) = \mathrm{hiP}(p)$, $\mathrm{lo}(p) = \mathrm{loS}(p)$.
According to Definition 3.4, $\mathrm{hiP}(p)$ is a vertex $q' \in R$ such that $\mathrm{Backw}(q') > p$, and $\mathrm{loS}(p) = \mathrm{Backw}(p')$ for some vertex $p' \in L_{<p}$. Thus, $\mathrm{Backw}(p') > q'$ and $\mathrm{Backw}(q') > p > p'$, a contradiction to Lemma 3.3, condition (i), first alternative.

Case 2: $\mathrm{hi}(p) = \mathrm{hiS}(p)$, $\mathrm{lo}(p) = \mathrm{loS}(p)$.
Here, $\mathrm{hiS}(p) = \mathrm{Forw}(p')$ for some vertex $p' \in L_{>p}$ and $\mathrm{loS}(p) = \mathrm{Backw}(p'')$ for some vertex $p'' \in L_{<p}$. Thus, $\mathrm{Backw}(p'') > \mathrm{Forw}(p')$ and $p' > p > p''$, a contradiction to condition (ii).

Case 3: $\mathrm{hi}(p) = \mathrm{hiP}(p)$, $\mathrm{lo}(p) = \mathrm{loP}(p)$.
Now, $\mathrm{hiP}(p)$ is a vertex $q' \in R$ such that $\mathrm{Backw}(q') > p$, $\mathrm{loP}(p)$ is a vertex $q'' \in R$ such that $\mathrm{Forw}(q'') < p$. Thus, $q'' > q'$ and $\mathrm{Backw}(q') > p > \mathrm{Forw}(q'')$, a contradiction to condition (iii).

The fourth case is symmetric to Case 1, the contradiction is against condition (i), second alternative.

After the shooting queries are answered, the running time of the other operations is linear in n. □

4. Straight Counter-Walks

In this section we consider a problem that seems to be symmetric to the one we solved in the previous section. We are still given the polygon P with two marked vertices, s and g, and two guards have to walk along two chains, L and R, of the polygon such they are always mutually visible. But this time, one of the guards starts at s and goes to g while the other one starts at g and walks to s, still without backtracking. We call this type of walk a *counter-walk*. Though the results for straight counter-walks are to some extent similar to those for straight walks, a direct duality does not seem to exist.

Throughout this section, we assume that L and R are weakly visible and that s and g are mutually visible, which is necessary for the existence of a counter-walk.

Definition 4.1

(i) A *counter-walk* on L, R is defined just like a walk (Definition 2.2), but with the boundary conditions $l(0) = s$, $l(1) = g$, $r(0) = g$, $r(1) = s$.

(ii) A *straight* counter-walk has a non-decreasing function l and a non-increasing function r with respect to the orientation of L and R.

(iii) P is called *(straight) counter-walkable* if it admits a (straight) counter-walk.

(iv) A *straight counter-walk on P from line segment $\overline{p_0\ q_0}$ to $\overline{p_1\ q_1}$* in P, where $p_0 < p_1$ and $q_1 < q_0$, is required to fulfill the conditions $l(0) = p_0$, $r(0) = q_0$, $l(1) = p_1$, and $r(1) = q_1$; and l is non-decreasing and r is non-increasing.

Lemma 4.2 *There is no straight counter-walk for L, R, if one of the following applies, see Figure 12.*

(i) $\exists p \in L, q \in R$: $\left(q > \mathrm{Forw}(p) \in R \quad \textit{and} \quad p < \mathrm{Backw}(q) \in L\right)$ *or* $\left(q < \mathrm{Backw}(p) \in R \quad \textit{and} \quad p > \mathrm{Forw}(q) \in L\right)$

(ii) $\exists p, p' \in L$: $p < p' \quad \textit{and} \quad R \ni \mathrm{Forw}(p) < \mathrm{Backw}(p') \in R$

(iii) $\exists q, q' \in R$: $q < q' \quad \textit{and} \quad L \ni \mathrm{Forw}(q) < \mathrm{Backw}(q') \in L$

In case (i) not even a counter-walk exists.

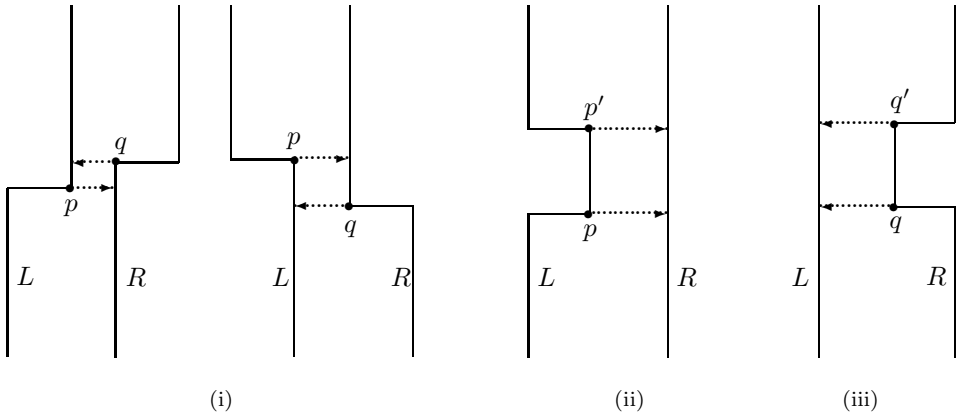


Fig. 12. Cases where no straight counter-walks exist (Lemma 4.2)

Proof. Assume that the first alternative of situation (i) applies. Let p' denote the first walk partner of $\mathrm{Succ}(q)$ with respect to the orientation of chain L, and let q' be the first walk partner of $\mathrm{Pred}(p)$. It follows from the assumption that $\mathrm{Pred}(p) < \mathrm{Backw}(q) \leq p'$ and, symmetrically, $\mathrm{Succ}(q) > \mathrm{Forw}(p) \geq q'$.

It follows that a counter-walk (which is a continuous function) would have to go through $\mathrm{Pred}(p)$ *before* reaching $\mathrm{Succ}(q)$ for the first time as well as it would have to pass $\mathrm{Succ}(q)$ before reaching $\mathrm{Pred}(p)$ for the first time, a contradiction, see Figure 12 (i).

In situation (ii), let q and q' denote the walk partners of $\mathrm{Pred}(p)$ and $\mathrm{Succ}(p')$ in a prospective straight counter-walk. Then $q' \geq \mathrm{Backw}(p') > \mathrm{Forw}(p) \geq q$, contradicting $\mathrm{Pred}(p) < \mathrm{Succ}(p')$. The other situation can be dealt with analogously, see Figure 12 (ii) and (iii). □

While the necessity of the conditions of Lemma 4.2 for the existence of a straight counter-walk is clear, sufficiency is true but not evident. We proceed as in Section 3 by computing, for each vertex $v \in L$, the interval(!) of its possible counter-walk partners on R.

First, we define an interval $[\text{C-lo}(v), \text{C-hi}(v)]$ that captures the restrictions imposed by the Forw and Backw shots. Unlike the case of straight walks, where the analogue intervals $[\mathrm{lo}(v), \mathrm{hi}(v)]$ are already the walk intervals, here the intervals $[\text{C-lo}(v), \text{C-hi}(v)]$ may be partially invisible from their associated vertex v. We will

see that the real counter-walk interval for a vertex v is the interval $[\text{C-lo}(v), \text{C-hi}(v)]$ intersected with $\text{vis}(v)$, the parts of P visible from v.

Definition 4.3 For a vertex $p \in L$ we define:

$$\begin{aligned}
\text{C-hiS}(p) &:= \min\{\text{Forw}(p') \in R \mid p' \text{ vertex in } L_{\leq p}\} \\
\text{C-hiP}(p) &:= \min\{\text{Pred}(q) \mid q \text{ vertex in } R \text{ and } L \ni \text{Forw}(q) < p\} \\
\text{C-hi}(p) &:= \min\{\text{C-hiS}(p), \text{C-hiP}(p), g\} \\
\text{C-loS}(p) &:= \max\{\text{Backw}(p') \in R \mid p' \text{ vertex in } L_{\geq p}\} \\
\text{C-loP}(p) &:= \max\{\text{Succ}(q) \mid q \text{ vertex in } R \text{ and } L \ni \text{Backw}(q) > p\} \\
\text{C-lo}(p) &:= \max\{\text{C-loS}(p), \text{C-loP}(p), s\}
\end{aligned}$$

These quantities are defined analogously for vertices $q \in R$. As in Section 3, the letters S and P stand for *self* and *partner*, correspondingly, meaning that the restriction is caused by a vertex on the same chain or by a vertex on the opposite chain. The letter C is for *counter*, as opposed to straight.

Lemma 4.4 *Each possible walk partner for a vertex v in a straight counter-walk is contained in the interval* $[\text{C-lo}(v), \text{C-hi}(v)]$.

The proof is obvious.

Lemma 4.5 *Let $p \in L$, $q \in R$ be two vertices of P.*

(i) If $q < \text{C-lo}(p)$ then $p < \text{C-lo}(q)$, if $q > \text{C-hi}(p)$ then $p > \text{C-hi}(q)$.

(ii) $p \in [\text{C-lo}(q), \text{C-hi}(q)] \iff q \in [\text{C-lo}(p), \text{C-hi}(p)]$.

Proof.

(i) If $q < \text{C-lo}(p) = \text{C-loP}(p)$ then, according to the definition, $\text{Pred}(\text{C-loP}(p))$ is the maximum vertex $q' \in R$ with $\text{Backw}(q') > p$. Because of $q \leq q'$, if follows from the definition of C-loS that $\text{Backw}(q') \leq \text{C-loS}(q)$. Thus, $p < \text{Backw}(q') \leq \text{C-loS}(q) \leq \text{C-lo}(q)$.
If $q < \text{C-lo}(p) = \text{C-loS}(p)$ then there is a vertex $p' \geq p$ such that $\text{C-loS}(p) = \text{Backw}(p')$. According to the definition of C-loP, $p \leq p' < \text{C-loP}(q) \leq \text{C-lo}(q)$.
The second assertion of (i) is symmetric to the first.

(ii) This is a direct consequence of (i).

□

From now on we assume that none of the conditions of Lemma 4.2 applies.

We have to prove that P admits a counter-walk. A major step towards this is proving the following two lemmata.

Lemma 4.6 *For all vertices v of P we have* $[\text{C-lo}(v), \text{C-hi}(v)] \neq \emptyset$.

Proof. Assume there is a vertex p of L with $\text{C-hi}(p) < \text{C-lo}(p)$. This immediately implies $\text{C-lo}(p) \neq s$, $\text{C-hi}(p) \neq g$.

Case 1: $\text{C-hi}(p) = \text{C-hiP}(p) < \text{C-lo}(p) = \text{C-loP}(p)$

According to Definition 4.3, $\text{Pred}(\text{C-loP}(p))$ is a vertex $q \in R$ with $\text{Backw}(q) > p$, and $\text{Succ}(\text{C-hiP}(p))$ is a vertex $q' \in R$ such that $\text{Forw}(q') < p$, $q' < q$. But this is exactly situation (iii) of Lemma 4.2.

Case 2: $\text{C-hi}(p) = \text{C-hiS}(p) < \text{C-lo}(p) = \text{C-loP}(p)$

Here, $\text{Pred}(\text{C-loP}(p)) = q$ just as in Case 1, and there is a vertex $p' \in L$, $p' \leq p$

with Forw(p') = C-hiS(p) $\le q$ and equality is impossible due to the general position assumption. We have Backw(q) $> p \ge p'$ and Forw(p') $< q$; this is situation (i) of Lemma 4.2.

Case 3: C-hi(p) = C-hiP(p) < C-lo(p) = C-loS(p)
This becomes Case 2 if we reverse the order on L and R and exchange s and g.

Case 4: C-hi(p) = C-hiS(p) < C-lo(p) = C-loS(p)
There are two different vertices p', $p'' \in L$ with $p' \le p \le p''$ and Forw(p') = C-hiS(p) < C-loS(p) = Backw(p''). This is situation (iii) of Lemma 4.2 (with L and R exchanged). □

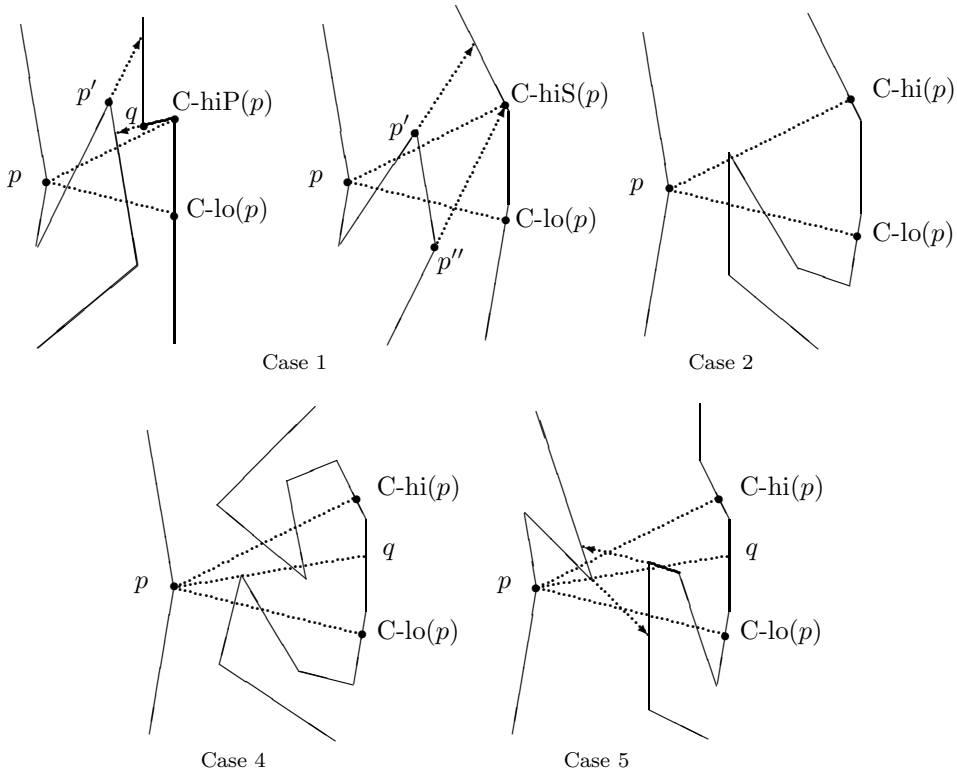


Fig. 13. Lemma 4.7

Lemma 4.7 *For all vertices v of P we have* [C-lo(v), C-hi(v)] $\cap$ vis(v) $\neq \emptyset$.

Proof. We already know by Lemma 4.6 that [C-lo(v), C-hi(v)] $\neq \emptyset$ holds for all vertices v. Assume that [C-lo(p), C-hi(p)] is completely invisible from a vertex $p \in L$. We distinguish the main cases, all others are symmetric; see Figure 13. Note that cases in which some part of [C-lo(v), C-hi(v)] intersects $\overline{p \text{ C-lo}(p)}$ or $\overline{p \text{ C-hi}(p)}$ do not need to be considered. That interval must be hidden behind something else than itself since it is assumed to be invisible from p.

Case 1: Some part of $L_{<p}$ intersects $\overline{p \text{ C-hi}(p)}$.
Then there is a vertex $p' < p$ with Backw(p') > C-hi(p) such that $(p', p]$ is invisible from C-hi(p).

If C-hi(p) = C-hiP(p) then there is a vertex $q \in R$, $q <$ Backw(p') with Forw(q) $< p'$; this is the forbidden situation (i) of Lemma 4.2.
If C-hi(p) = C-hiS(p) then there is a vertex $p'' \leq p'$ with Forw(p'') = C-hi(p) $<$ Backw(p'); this is situation (ii) of Lemma 4.2.

Case 2: Assume that Case 1 does not apply and that some part of $R_{<\text{C-lo}(p)}$ intersects $\overline{p\ \text{C-hi}(p)}$.
Then $L_{\leq p}$ would be invisible from C-hi(p), but the definition of C-hi tells us otherwise (note that in case C-hi(p) = g point s is visible by assumption).

For the remaining cases we assume that neither Case 1 nor Case 2 applies.

Case 3: $L_{<p}$ intersects $\overline{p\ \text{C-lo}(p)}$ and $L_{>p}$ intersects $\overline{p\ \text{C-hi}(p)}$ such that together they completely obstruct the view from p to [C-lo(v), C-hi(v)].
Then p is invisible from R.

Case 4: $R_{<\text{C-lo}(p)}$ intersects $\overline{p\ \text{C-lo}(p)}$ and $R_{>\text{C-hi}(p)}$ intersects $\overline{p\ \text{C-hi}(p)}$ such that together they completely obstruct the view from p to [C-lo(v), C-hi(v)].
As in Case 3, there is a point $q \in$ [C-lo(v), C-hi(v)] that is invisible from L.

Case 5: $R_{<\text{C-lo}(p)}$ intersects $\overline{p\ \text{C-lo}(p)}$ and $L_{>p}$ intersects $\overline{p\ \text{C-hi}(p)}$ such that together they completely obstruct the view from p to [C-lo(v), C-hi(v)].
Now, there is a point $q \in$ [C-lo(v), C-hi(v)] such that $\overline{pq}$ intersects both, $R_{<\text{C-lo}(p)}$ and $L_{>p}$. One of the situations (i) of Lemma 4.2 applies, depending on which of $R_{<\text{C-lo}(p)} \cap \overline{pq}$ or $L_{>p} \cap \overline{pq}$ comes first on the ray $\overrightarrow{pq}$. □

Lemma 4.8 *The set* [C-lo(v), C-hi(v)] ∩ vis(v) *is connected for all vertices* v *of* P.

Proof. Assume $q, q'' \in$ [C-lo(p), C-hi(p)] ∩ vis(p), $q < q' < q''$, q' invisible from p, for some vertex $p \in L$.

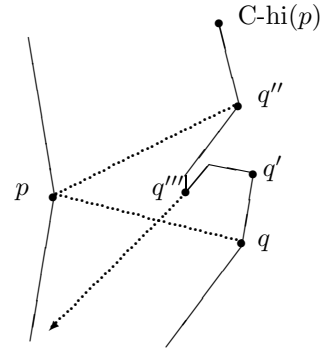


Fig. 14. Lemma 4.8; Forw(q''') $< p$ contradicts the definition of C-hi(p).

Then $\overline{p\ q'}$ must be intersected by some part of $[q, q'']$, resulting in a vertex $q''' \in [q, q'']$ with either Forw(q''') $< p$ or Backw(q''') $> p$, but this is impossible by the definition of C-hi and C-lo; see Figure 14. □

Now that we know that the visible part of [C-lo(v), C-hi(v)] is an interval, it is convenient to give names to its endpoints.

Definition 4.9 For a vertex v of P we define:

$$\begin{aligned} \text{VC-lo}(v) &:= \min\big([\text{C-lo}(v), \text{C-hi}(v)] \cap \text{vis}(v)\big), \\ \text{VC-hi}(v) &:= \max\big([\text{C-lo}(v), \text{C-hi}(v)] \cap \text{vis}(v)\big). \end{aligned}$$

Lemma 4.10 *Considered as functions* $\{\text{vertices of } L\} \to R$ *or* $\{\text{vertices of } R\} \to L$, C-hi *and* C-lo *are monotonically decreasing. The same property holds for* VC-hi *and* VC-lo.

Proof. The property for C-hi and C-lo follows immediately from the definition.

Now, take two vertices $p,\ p' \in L,\ p < p'$. We have to show that VC-lo$(p) \geq$ VC-lo(p'). If VC-lo(p') = C-lo(p') then the assertion is true because VC-lo$(p) \geq$ C-lo(p) by definition and C-lo is monotonic.

In the remaining case VC-lo(p') > C-lo(p') we assume VC-lo(p) < VC-lo(p'). The segment $\overline{p'\ \text{VC-lo}(p')}$ must be touched from below, so that VC-lo(p') is the lowest point above C-lo(p') that is visible from p', due to Lemma 4.8. We distinguish two subcases.

Case 1: $[p, p')$ touches $\overline{p'\ \text{VC-lo}(p')}$, thereby intersecting $\overline{p'\ \text{C-lo}(p')}$.
Then there is a vertex $p'' \in (p, p')$ with Backw$(p'') \geq$ VC-lo(p') > C-lo(p) in contradiction to the maximality property of C-lo(p).

Case 2: $[\text{VC-lo}(p), \text{VC-lo}(p')]$ touches $\overline{p'\ \text{VC-lo}(p')}$.
Then there would be a vertex $q \in [\text{VC-lo}(p), \text{VC-lo}(p')\,)$ that is visible from p', a contradiction, because of C-lo(p') < q.

Since no part of $L_{<p} \cup R_{<\text{VC-lo}(p)}$ can reach $\overline{p'\ \text{VC-lo}(p')}$ without crossing the visibility segment $\overline{p\ \text{VC-lo}(p)}$, a contradiction follows.

Similarly, one shows that VC-hi(p) is decreasing. □

Lemma 4.11 *Let* $p \in L$, $q \in R$ *be two vertices of* P.

(i) If $q <$ VC-lo(p) *then* $p <$ VC-lo(q)*; if* $q >$ VC-hi(p) *then* $p >$ VC-hi(q).

(ii) $p \in [\text{VC-lo}(q), \text{VC-hi}(q)] \iff q \in [\text{VC-lo}(p), \text{VC-hi}(p)]$.

Proof.

(i) Assume $q <$ VC-lo(p). If $q <$ C-lo(p) then $p <$ C-lo$(q) \leq$ VC-lo(q) by Lemma 4.5.

For the remaining case, C-lo$(p) \leq q <$ VC-lo(p), we know that VC-lo(p) is visible but C-lo(p) is invisible from p, and $\overline{p\ \text{VC-lo}(p)}$ must be touched "from below". We distinguish two subcases.

Case 1: $L_{<p}$ touches $\overline{p\ \text{VC-lo}(p)}$, thereby intersecting $\overline{p\ \text{C-lo}(p)}$.

Here, there is a vertex $p' \in L$ with Backw$(p') >$ VC-lo$(p) > q$. According to the definition, $p' <$ C-loP$(q) \leq$ C-lo(q). The interval $(p', p]$ is invisible from q, so $p <$ VC-lo(q).

Case 2: $R_{<\text{VC-lo}(p)}$ touches $\overline{p\ \text{VC-lo}(p)}$

Now, there is a vertex $q' <$ VC-lo(p) with Backw$(q') > p$, q' visible from p. The connectedness of $[\text{C-lo}(p), \text{C-hi}(p)] \cap \text{vis}(p)$ implies $q' <$ C-lo$(p) \leq q$, thus $L_{\leq p}$ is invisible from q and $p <$ VC-lo(q).

The other assertion is symmetrical.

(ii) This is a direct consequence of (i).

□

We need to compute the sets $[\text{VC-lo}(v), \text{VC-hi}(v)] = [\text{C-lo}(v), \text{C-hi}(v)] \cap \text{vis}(v)$ (still assuming that none of the situations of Lemma 4.2 applies). As in Section 3, after performing all Forw and Backw shots, all intervals $[\text{C-lo}(v), \text{C-hi}(v)]$ can be computed in linear time. But computing vis(v) and intersecting it with

[C-lo(v), C-hi(v)] for all vertices may require as much as $\Omega(n)$ operations per vertex, and is therefore much too expensive. However, Lemma 4.7 and Lemma 4.8 show that [VC-lo(v), VC-hi(v)] is a nonempty interval included in [C-lo(v), C-hi(v)], so we do not really need to compute all of vis(v).

Lemma 4.12 *All intervals* [VC-lo(v), VC-hi(v)] *can be computed in time* $O(n \log n)$.

Proof. The idea is to use the optimal shortest path queries provided by Guibas and Hershberger[7]. Given a triangulated polygon, after a linear time preprocessing step their algorithm determines in time $O(\log n)$ the length of the shortest path inside a polygon P between two arbitrary points. This algorithm can be modified to report the *first* segment of such a shortest path within the same time. For our purposes, an $O(n \log n)$ triangulation method (as described by Preparata and Shamos[9]) is sufficient.

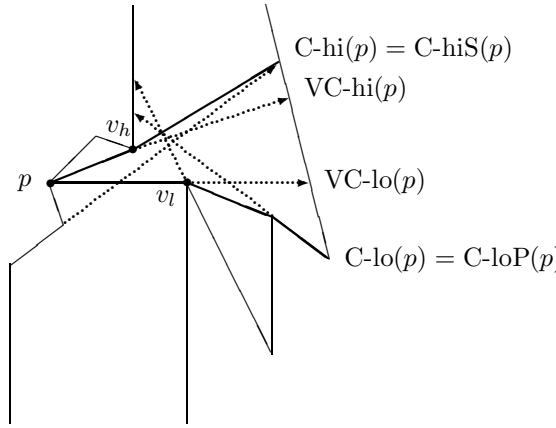


Fig. 15. An illustration of Lemma 4.12

Consider for a vertex $p \in L$ the shortest paths from p to C-lo(p) and to C-hi(p). Let v_l and v_h denote the first vertices after p on the two shortest paths. Then VC-lo(p) and VC-hi(p) are the hit points of the shots from p through v_l and v_h; see Figure 15.

In this way, we can compute each point VC-lo(v), VC-hi(v) from the points C-lo(v), C-hi(v) by one shortest path query and one shooting query each, in an overall running time of $O(n \log n)$. □

Now we have the tools for deciding if a straight counter-walk exists. The next theorem gives us both an equivalent criterion and a method for efficiently computing such a walk.

Theorem 4.13 *There is a straight counter-walk iff the chains L and R are mutually weakly visible and none of the conditions of Lemma 4.2 applies. Such a straight counter-walk can be computed in* $O(n \log n)$ *time.*

Proof. Necessity was shown in Lemma 4.2.

In order to prove sufficiency, we explicitly construct a straight counter-walk. From Lemma 4.7 we know that for all vertices v the interval [VC-lo(v), VC-hi(v)] is

not empty and is visible from v, and that all possible walk partners for a vertex v must be contained in that interval.

From Lemma 4.10 we know that VC-lo and VC-hi are monotonically decreasing. If we assign and mark the point VC-lo(p) as walk partner to each vertex $p \in L$ then these walk partners are in the correct order for a straight counter-walk. (In fact, every two walk lines cross. This corresponds to the property of straight walks that no two walk lines cross, see Lemma 2.3.)

A vertex $q \in R$ that has not yet been assigned a walk partner lies between two nearest marked points, VC-lo(p) and VC-lo(p'), where $p > p'$. It follows from Lemma 4.11 that $p' <$ VC-lo(q) $\leq p$. So we can choose VC-lo(q) as a walk partner for q. Still all walk partners are mutually visible and in the correct order for a straight counter-walk.

Now that all vertices have been assigned walk partners, a straight counter-walk can be obtained in the following way. Assume that l and r, the guards on L and R, are at the marked points p_i and q_i, respectively, and let p_{i+1} and q_{i-1} denote the next marked points on L and R. Then the guards proceed in such a way that their walk segment $\overline{lr}$ rotates about the intersection point $\overline{p_iq_i} \cap \overline{p_{i+1}q_{i-1}}$. This results in the type of elementary walk instruction depicted in Figure 5 (ii). □

By analogy to the case of straight walks, we have the following corollary.

Corollary 4.14 *If P admits a straight counter-walk then for every vertex v each point of the interval* [VC-lo(v), VC-hi(v)] *is a suitable walk partner for v.*

5. General Walks

Let us now allow backtracking of the guards. We will see in this section that a polygon admits a general walk if and only if it is weakly visible and contains no deadlocks, see Figure 8, whereas wedges are permitted, in contrast to straight walkable polygons. All straight walks for a given polygon are of equal length, but with backtracking allowed it makes sense not only to ask if a given polygon is walkable and to construct a walk, but also to ask for a *shortest* walk.

Our algorithm for the general walk problem consists of the following steps. First, we solve the shooting problem of Section 3, build the search structures for all vertices and hit points, and check for weak visibility of L and R and for the deadlock conditions described in Lemma 3.3 (i). We give an algorithm *turnpoint* to compute the necessary and sufficient turning points for a walk. With that information we can partition P into straight walkable and straight counter-walkable pieces and apply the results of Section 3 and Section 4 for each piece. We can show that the concatenation of these walks is of minimal length and that the whole algorithm runs in time $O(n \log n + k)$, where n is the number of vertices in P and k is the number of walk instructions the minimal walk consists of.

Once the weak visibility of L and R is verified with the help of Lemma 3.2, those shots Forw(p) or Backw(p) of some vertex $p \in L$ that hit L are no longer interesting. From now on we assume that *L and R are weakly visible.* Also we use the following convention. Whenever two shots Forw(p) and Backw(p') are compared,

where p, $p' \in L$, then these hit points are assumed to lie on R, and vice versa. In Lemma 3.3 (i) we have seen that in the case of a deadlock situation, as shown in Figure 8 (i), not even a general walk exists; thus we assume from now on that *there are no deadlocks in* P.

Obviously, a wedge $W = [p, p']$ on L, as shown in Figure 8 (ii), causes the guard on R to turn, backtrack, and turn again, while its partner moves through W. The problem is how to organize the backtrack moves on R. The example polygon depicted in Figure 16 shows that not every wedge on L defines turning points on R.

The following algorithm *turnpoint* scans L from s to g and reports the turning points on R that are necessary for a walk.

Algorithm *turnpoint*

```
1    utp := s;
2    ltp := g;
3    FOR all reflex vertices p ∈ L in increasing order DO
4         IF ltp > utp
5         THEN IF Backw(p) > utp
6                   THEN utp := Backw(p)
7                   ELSE IF Forw(p) < utp
8                             THEN ltp := Forw(p)
                              ENDIF
                    ENDIF
9         ELSE  IF Forw(p) < ltp
10                  THEN ltp := Forw(p)
11                  ELSE IF Backw(p) > ltp
12                            THEN report-turning-points(utp,ltp);
13                                 utp := Backw(p);
14                                 ltp := g
                              ENDIF
                    ENDIF
          ENDIF
     ENDFOR;
15   IF ltp < utp
16   THEN report-turning-points(utp,ltp)
     ENDIF
```

The algorithm *turnpoint* scans L and stops at every reflex vertex (line 3). It keeps track in line 5–6 of the maximum (with respect to the order on R) Backw shot *utp* because this is a candidate for an upper turning point. In Figure 16, *utp* takes on the values q_4, then q_6. Maximization of *utp* stops once the first Forw shot $<$ *utp* is detected in line 7, namely point q_2 in Figure 16. This point is called *ltp* because it gives rise to a candidate for a lower turning point (line 8). At this moment, a *wedge* has been detected, and *utp* is a necessary upper turning point, whereas *ltp* is still to be minimized. Note that Forw$(p) <$ *utp* implies Backw$(p) <$ *utp*, so line 7 is in fact executed each time the condition is fulfilled. Afterwards, *ltp* $<$ *utp* holds.

As we proceed scanning L, two cases can occur. First, a shot Forw(p) is encountered that is less than the current minimum Forw shot *ltp* (line 9). This happens in

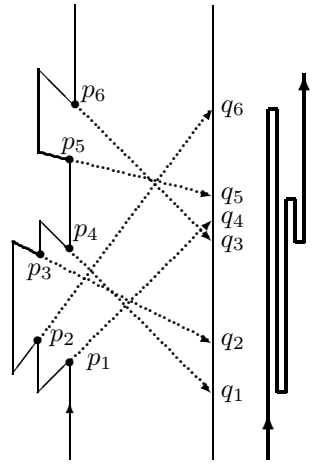


Fig. 16. A walk with backtracking

Figure 16 on reaching vertex p_4. At this moment, *utp* equals q_6, and *ltp* equals q_2, and it is clear that the guard on R has to go back (at least) to $q_1 = \text{Forw}(p_4)$ after visiting q_6. Therefore, q_1 becomes the lower turning point *ltp* (line 10).

The second possible event is that a shot Backw(p) greater than *ltp* occurs (line 11), for example point q_5 in Figure 16. Again, Forw(p) must be higher than *ltp* in this case. Then the guard on R must turn again at *ltp* and walk at least to Backw(p). Therefore, (*utp*, *ltp*) define a pair of consecutive turning points on R and have to be reported as such (line 12). The point Backw(p) is a new candidate for an upper turning point (line 13), a corresponding lower turning point of which is still to be detected, if it exists. As long as this is the case, *ltp* is set to g (line 14 and line 2). Initially, *utp* is set to s (line 1) to denote that no candidate for an upper turning point has so far been detected.

After leaving the *FOR* loop (line 3–14), we have to check if a new wedge has been detected in line 7 that has not yet been reported in line 12. If so, we report this last wedge and terminate (line 15–16).

A symmetrical version of the algorithm can be used to compute turning points on L.

The above discussion shows that any walk for P has to visit the upper and lower turning points in the order given by the algorithm. In the rest of this section we will prove that there is indeed a walk that turns at exactly these points. Such a walk is clearly of minimal length.

Definition 5.1 Let $utp = \text{Backw}(up)$, $ltp = \text{Forw}(lp)$ be a pair of points that has been reported by algorithm *turnpoint*.
If $up, lp \in L$ (and $utp, ltp \in R$) we call (up, lp, utp, ltp) a *left maximum wedge*. Otherwise, i.e. if $up, lp \in R$, we call (utp, ltp, up, lp) a *right maximum wedge*. (Note that we use the order inside the 4-tuples to distinguish between left and right maximum wedges.) In either case, such a 4-tuple is called a *maximum wedge*.

The set of maximum wedges is totally ordered in the following sense. Two

maximum wedges on the same side are in the order in which they were reported by the algorithm. Two maximum wedges on opposite sides can always be separated by a line segment from L to R, due to the following lemma. Thus, we may speak of the first, second, last, etc. maximum wedge. We will use the sign $\prec$ for this order.

Lemma 5.2 *For two maximum wedges $W_1 = (utp_1, ltp_1, up_1, lp_1)$ and $W_2 = (ut_2, lt_2, utp_2, ltp_2)$ on L and R, resp., the following holds.*

$$\begin{array}{ll} \textit{Either} & \Big(utp_1 < up_2 \textit{ and } lp_1 < ltp_2\Big) \\ \textit{or} & \Big(lp_2 < ltp_1 \textit{ and } utp_2 < up_1\Big). \end{array}$$

In the first case, we denote this by $W_1 \prec W_2$, in the second by $W_2 \prec W_1$.

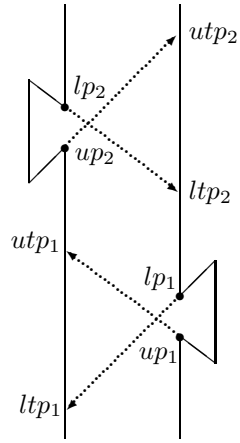


Fig. 17. Maximum wedges on opposite sides do not interfere.

Proof. Otherwise a deadlock situation like Figure 8 (i) would occur, see Figure 17. □

By the next two lemmata we show that a straight walk exists between two consecutive maximum wedges, and that a straight counter-walk exists inside a maximum wedge. In a sense, these are generalizations of Theorem 3.11 and Theorem 4.13 in that they solve the corresponding *edge-to-edge problem* instead of the *point-to-point problem*.

Lemma 5.3 *Let $W_1 \prec W_2$ be two consecutive maximum wedges (in the order explained above). Then there is a straight walk on P from $\overline{lp_1\ ltp_1}$ to $\overline{up_2\ utp_2}$.*

Proof. We distinguish two cases. W_1 and W_2 can be on the same side or on opposite sides of P.

Assume that $W_1 = (up_1, lp_1, utp_1, ltp_1)$ and $W_2 = (up_2, lp_2, utp_2, ltp_2)$ are both *left* maximum wedges, see Figure 18 (i). Then $ltp_1 < utp_2$ and $lp_1 < up_2$, because these two maximum wedges were reported consecutively by our algorithm, so it makes sense to speak of a straight walk from $\overline{lp_1\ ltp_1}$ to $\overline{up_2\ utp_2}$, both of which are visibility segments.

Let us consider the straight walk problem for the polygon P' consisting of the two chains $L' = \overline{ltp_1\ lp_1}[lp_1, up_2]\overline{up_2\ utp_2}$ and $R' = [ltp_1, utp_2]$ with $s' = ltp_1$ and

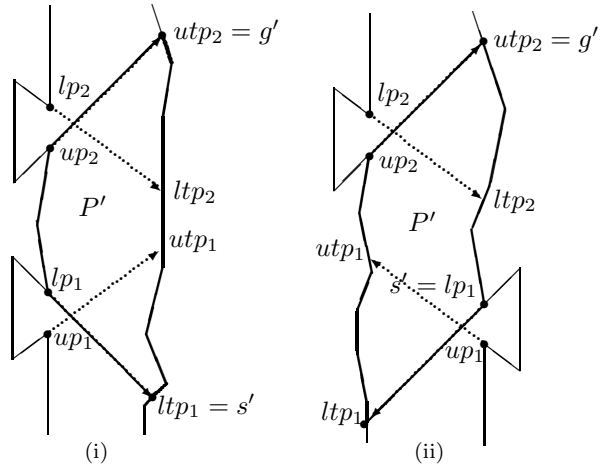


Fig. 18. Lemma 5.3

$g' = utp_2$. All of ltp_1, lp_1, up_2 and utp_2 are non-reflex vertices in P', so each reflex vertex of P' is also a reflex vertex of P. From the assumption that L and R are weakly visible it follows that each point of R' is visible from one point of L', and, since the two wedges are maximum, there is no vertex $p \in L'$ with $\mathrm{Backw}(p) > utp_2$ resp. $\mathrm{Forw}(p) < ltp_1$. Now Lemma 3.2 shows that each point of L' is visible from one point of R'; thus L' and R' are weakly visible.

There is no deadlock, see Figure 8 (i), in P', since otherwise a deadlock in P would exist, or weak visibility of P would be violated if a shot hits the opposite chain in P' but not in P.

Assume that P' contains a wedge, see Figure 8 (ii), defined by the reflex vertices $v < v'$, i.e. $\mathrm{Forw}(v') < \mathrm{Backw}(v)$, where w.l.o.g. $[v, v']$ does not contain another pair of vertices with this property. Due to weak visibility, this must be a wedge in P, too. Because it is minimal in the above sense, $[v, v']$ must be contained in a maximum wedge W reported by the algorithm, which must fulfill $W_1 \prec W \prec W_2$ (or a violation of Lemma 5.2 must occur in case W is a right maximum wedge and overlaps with one of W_1, W_2). This contradicts our assumption that W_1 and W_2 are consecutive maximum wedges, so there is no wedge in P'. Now Theorem 3.11 implies the existence of a straight walk in P' from s' to g'.

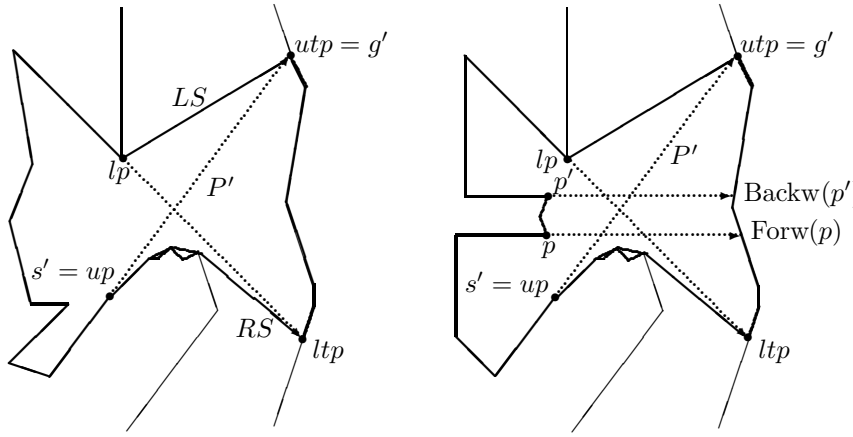
It remains to show that there exists a straight walk from edge $\overline{lp_1\ ltp_1}$ to edge $\overline{up_2\ utp_2}$. There is no vertex $q \in R'$ with $\mathrm{Backw}(q) > up_2$ because there is no deadlock in P, so $\mathrm{hi}'(up_2) = g' = utp_2$ (hi' denotes hi with respect to P'). Symmetrically, we have $\mathrm{lo}'(lp_1) = s' = ltp_1$. Following the proof of Lemma 3.9, there is a straight walk for P' in which each vertex $p \in L'$ has $\mathrm{lo}'(p)$ as its walk partner. From $\overline{up_2, \mathrm{lo}'(up_2)}$ on, the left guard may stand still while the right one proceeds to $g' = utp_2 = \mathrm{hi}'(up_2)$. Lemma 3.8 guarantees that during the last move the guards remain covisible. Consequently, there is a walk in P' (and also in P) from $\overline{lp_1\ ltp_1}$ to $\overline{up_2\ utp_2}$.

For the second case, i.e. W_1 and W_2 are on opposite sides, we assume w.l.o.g.

that $W_1 = (utp_1, ltp_1, up_1, lp_1)$ and $W_2 = (up_2, lp_2, utp_2, ltp_2)$, see Figure 18 (ii). Then $ltp_1 < utp_1 < up_2$ and $lp_1 < ltp_2 < utp_2$ by Lemma 5.2, and the polygon P' consisting of $L' = \overline{lp_1\ ltp_1}[ltp_1, up_2]\overline{up_2\ utp_2}$ and $R' = [lp_1, utp_2]$ with $s' = lp_1$ and $g' = utp_2$ is well defined.

The rest of the proof for the second case differs from the first case only in the following arguments. The maximality of utp_2 still guarantees that there is no vertex $p \in L'$ with $\text{Backw}(p) > utp_2$. A vertex $p \in L'$ with $\text{Forw}(p) < lp_1$ would cause a deadlock in P. With Lemma 3.2 we have that L' and R' are weakly visible. We have $\text{hi}'(up_2) = g' = utp_2$ as before, and $\text{lo}'(ltp_1) = s' = lp_1$ because a vertex $q \in R'$ with $\text{Forw}(q) < ltp_1$ would violate the minimality of ltp_1. □

Lemma 5.4 *For a maximum wedge W, there is a straight counter-walk on P from $\overline{utp\ up}$ to $\overline{ltp\ lp}$.*



(i) A maximum Wedge. (ii) Not a maximum Wedge.

Fig. 19. Lemma 5.4

Proof. W.l.o.g. assume $W = (up, lp, utp, ltp)$ to be a left maximum wedge, see Figure 19 (i). Let LS denote the shortest path inside P from lp to utp and RS the shortest path from up to ltp. Both LS and RS are convex chains consisting of reflex vertices. We define a polygon P' with $L' = [up, lp]\, LS$ and $R' = RS\, [ltp, utp]$. We consider the straight counter-walk problem for P' with $s' = up$ and $g' = utp$. Let Forw' denote Forw with respect to P'.

The chains LS and RS are weakly visible, since the point of intersection of $\overline{utp\ up}$ and $\overline{ltp\ lp}$ can see both reflex chains, LS and RS. For all reflex vertices $q \in [ltp, utp]$ we have $\text{Forw}(q) > lp$ and $\text{Backw}(q) < up$ because otherwise there would be a deadlock in P. So each point in R' is visible from a point in L' by Lemma 3.2. On the other hand, for $p \in [up, lp]$ we have $\text{Forw}(p) \geq ltp$ due to the minimality of ltp and $\text{Backw}(p) \leq utp$ because of the maximality of utp, as corner points of a maximum wedge. Again from Lemma 3.2, it follows that each point of L' is visible from a point in R'. All together we have that L' and R' are weakly visible.

As we have seen, $\text{Forw}'(p) \geq ltp$ holds for all reflex vertices $p \in L'$. But there is

no $q \in R'_{\geq ltp}$ with $\text{Backw}'(q) \in L'$. By construction, we have $\text{Forw}'(q) > lp$ for all reflex vertices $q \in R'$. There is no $p \in L'_{>lp}$, i.e. $p \in LS$, with $\text{Backw}'(p) \in R'$. We conclude that condition (i) of Lemma 4.2 cannot apply.

With $\text{Forw}'(q) > lp$ and $\text{Backw}'(q') < lp$ for all $q, q' \in R'$ it follows that condition (iii) of Lemma 4.2 cannot apply, either.

Assume condition (ii) of Lemma 4.2 holds, i.e. $\text{Forw}'(p) < \text{Backw}'(p')$ for some $p < p'$. This could only be the case for $p, p' \in [up, lp]$ and $\text{Forw}'(p), \text{Backw}'(p') \in [ltp, utp]$, see Figure 19 (ii). But with $ltp < \text{Forw}'(p) < \text{Backw}'(p') < utp$ and $up < p < p' < lp$ it would follow that the algorithm *turnpoint* should have reported an additional pair of turning points.

We therefore know from Theorem 4.13 that there exists a straight counter-walk on P' from $\overline{up\ utp} = \overline{s'g'}$ to $\overline{utp\ up} = \overline{g's'}$. It remains to show that there is such a walk which takes on $\overline{ltp\ lp}$ as an intermediate position. The point ltp is visible from lp, and, by checking with Definition 4.3, one can verify that it is contained in $[\text{C-lo}'(lp), \text{C-hi}'(lp)]$, thus it is in $[\text{VC-lo}'(lp), \text{VC-hi}'(lp)]$. The existence of the walk in question follows from Corollary 4.14. □

Now the existence of straight resp. counter walk pieces between the turnpoints has been established; together, they form a walk for P. It remains to discuss the time performance. This is a subtle issue, for two reasons. First, consecutive maximum wedges on the same side can overlap. Second, we cannot afford to compute all Forw' and Backw' shots for each subpolygon P' occuring in the existence proof. Essentially, we have to live on the Forw and Backw shots computed for P in the beginning.

Lemma 5.5

(i) All straight walk pieces for P, i.e. all straight walks between two consecutive maximum wedges, can be computed in total time $O(n \log n + k)$.

(ii) All counter-walk pieces for P, i.e. all counter walks inside maximum wedges, can be computed in $O(n \log n + k)$ time, too.

Proof. We have already spent $O(n \log n)$ time for computing all Forw and Backw shots for P. The algorithm *turnpoint* reports all maximum wedges in time $O(n)$.

(i) For the straight walks between two maximum wedges (and in particular the first piece after s and the last piece towards g), our method of Section 3 applies, but with the following considerations.

Let P' be a subpolygon as defined in the proof of Lemma 5.3, see Figure 18, consisting of sides L' and R' with n' vertices. A straight walk from $\overline{lp_1\ ltp_1}$ to $\overline{up_2\ utp_2}$ can easily be obtained from a straight walk for P', see the proof of Lemma 5.3.

We make use of the fact that all Forw and Backw shots are already computed for P. All shooting queries for P' can be obtained in the following way, using total time $O(n')$. In Figure 18 (i), for example, in order to compute $\text{Forw}'(p)$ for a vertex $p \in L'$, we check if $\text{Forw}(p)$ belongs to $[lp_1, up_2] \cup [ltp_1, utp_2]$. If so, we have $\text{Forw}'(p) = \text{Forw}(p)$. Otherwise, $\text{Forw}'(p)$ is the point of intersection of $\overline{p\ \text{Forw}(p)}$ with either $\overline{lp_1\ ltp_1}$ or $\overline{up_2\ utp_2}$. Analogous observations hold for Backw shots and for vertices in R'.

The computation of hi′, lo′, etc. can be done in $O(n')$ time to obtain the desired walk. The sum of the n' is $O(k)$ although the straight walk pieces of P can overlap. Namely, if a vertex v belongs to t pieces like P' then v is visited $O(t)$ times in the minimal walk.

(ii) Given a left maximum wedge (up, lp, ltp, utp), we consider the subpolygon P' which was defined in the proof of Lemma 5.4, see Figure 19 (i). However, we do not compute the convex chains LS and RS, because we do not know in how many chains a vertex may appear (of course, the chains do restrict the visibility from $[up, lp]$ to $[ltp, utp]$). But the Forw and Backw shots from vertices in LS and RS do not restrict the walk from $\overline{up\ utp}$ to $\overline{lp\ ltp}$.
Let n' be the number of vertices of $[up, lp]$ and $[ltp, utp]$. We know from Lemma 5.4 that a straight counter-walk exists from $\overline{utp\ up}$ to $\overline{ltp\ lp}$.
In the proof of Lemma 5.4 we noted that for all reflex vertices $q \in [ltp, utp]$ we have $\text{Forw}(q) > lp$ and $\text{Backw}(q) < up$. Thus all Forw and Backw shots from $[ltp, utp]$ are irrelevant for our counter-walk.
Also following the proof of Lemma 5.4, for a reflex vertex $p \in [up, lp]$ we have $\text{Forw}(p) \geq ltp$ and $\text{Backw}(p) \leq utp$. So the shots from $[up, lp]$ are only interesting if they hit in $[utp, ltp]$, because $\text{Forw}(p) > utp$ and $\text{Backw}(p) < ltp$ are irrelevant restrictions for the intended counter-walk. Thus we do not need additional shooting queries for computing all intervals $[\text{C-lo}'(p), \text{C-hi}'(p)]$ for the vertices $p \in [up, lp]$, which can, therefore, be done in time $O(n')$.
After that, we compute the intervals $[\text{VC-lo}'(p), \text{VC-hi}'(p)]$ for the vertices $p \in [up, lp]$, using the method of shortest path queries and shooting queries described in Lemma 4.12. This requires time $O(\log n)$ per vertex, but the vertices of $[up, lp]$ are visited only once in our walk for P.

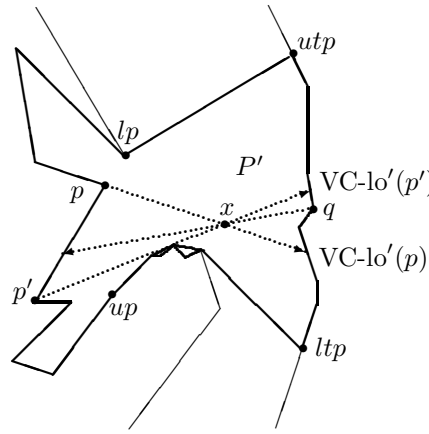


Fig. 20.

Now that we have all walk intervals for vertices on the left side, it is not difficult to assign all walk partners. As in the proof of Theorem 4.13, to each vertex $p \in [up, lp]$ we assign $\text{VC-lo}'(p)$ as walk partner. A vertex $q \in [ltp, utp]$ that has not yet been assigned a walk partner lies between two nearest assigned

points, VC-lo$'(p)$ and VC-lo$'(p')$, with $p > p'$. Let x denote the point of intersection of p VC-lo$'(p)$ and p' VC-lo$'(p')$. A suitable walk partner for q is the point of intersection of $\overline{p\,p'}$ and the ray $\overrightarrow{q\,x}$, since x can see the whole interval $[p', p]$, see Figure 20. All these walk partners can be assigned in time $O(n')$. Now all walk partners are mutually visible and in the correct order for a straight counter walk.

The overall time spent on this construction is $O(n \log n + k)$.

□

Theorem 5.6 *There is a walk for P iff the chains L and R are mutually weakly visible and P does not contain a deadlock, i.e. condition (i) of Lemma 3.3 does not hold. Furthermore, a walk of minimum length can be computed in time $O(n \log n + k)$ and $O(n)$ space where k is the number of walk instructions.*

Proof. The existence follows from Lemma 5.3 and Lemma 5.4, the time and space bounds from Lemma 5.5. □

6. The Lower Bound

In the preceding sections we have shown that one can, in time $O(n \log n)$, determine if a given simple polygon admits a walk from s to g, and compute a walk of minimum length in time $O(n \log n + k)$, if there is one. Here n denotes the number of edges of the polygon, and k is the size of the output, i.e. the number of walk instructions. As the next lemma shows, k can be quadratic.

Lemma 6.1 *The number k of walk instructions for the two guards is $\Theta(n^2)$ in the worst case.*

Proof. The polygon depicted in Figure 21 is walkable, but the guard on R has to make $\Omega(n^2)$ vertex visits on his way from s to g.

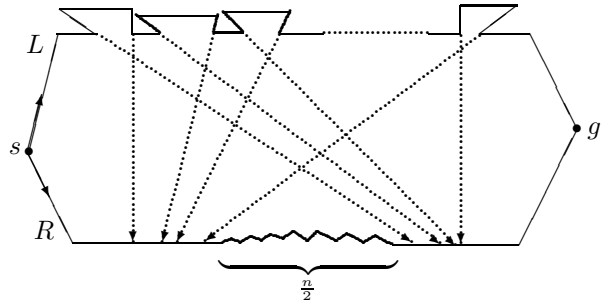


Fig. 21. Each of the $\frac{n}{2}$ vertices on R is visited $O(n)$ times.

On the other hand, the triangles, quadrilaterals, and rotations mentioned at the end of Section 5 are defined by a total of $O(n)$ points on P. Their number is $O(n^2)$ because they can only contain two *consecutive* points on the same chain. Since the walk is of minimum length, the guards do not take on the same relative position twice. □

Now we prove that our upper bound on $O(n \log n + k)$ for computing a walk is optimal. Since the walk instructions do in particular contain all turnpoints,

the assertion follows from the following theorem, whose proof is based on an idea communicated by Formann and Wagner[10].

Theorem 6.2 *Computing the turnpoints for a walkable polygon of n edges requires $\Omega(n \log n)$ time.*

Proof. Given n real numbers, $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_n < \pi$, in ascending order, and an unsorted sequence $\beta_1, \beta_2, \ldots, \beta_n$ of reals in (α_1, α_n), all different from the α_i, it takes $\Omega(n \log n)$ steps in the comparison model to determine for each β_i the unique $\alpha_{\sigma(i)}$ such that $\alpha_{\sigma(i)-1} < \beta_i < \alpha_{\sigma(i)}$ holds. We show that this problem can be reduced in linear time to the computation of turnpoints.

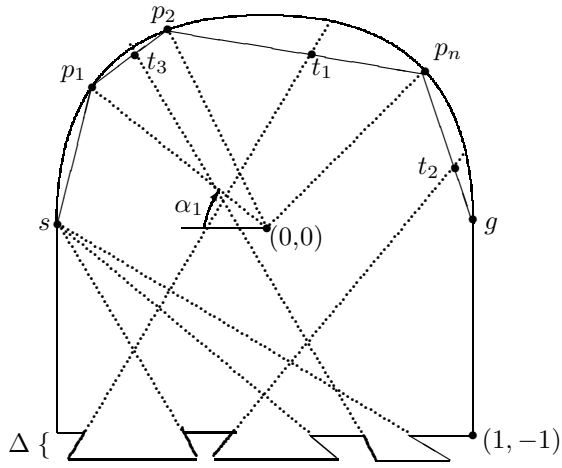


Fig. 22. The proof of the lower bound.

Let C be the upper half of the circle of radius 1 centered at $(0, 0)$, as shown in Figure 22. For an angle $\gamma \in (0, \pi)$ let $f(\gamma) := (-\cos(\gamma), \sin(\gamma))$ denote the corresponding point on C. Let $p_i := f(\alpha_i)$. We connect the (ordered!) points p_i by edges. Moreover, p_1 is connected with $(-1, 0)$, and p_n with $(1, 0)$. This polygonal chain will be the left side L of polygon P.

Next, we choose $\Delta > 0$ so small that the rectangular slab defined by $(-1, -1)$, $(-1, -1-\Delta)$, $(1, -1-\Delta)$, and $(1, -1)$, is spacious enough to accomodate n wedges all of whose right edges aim at s, whereas the left edge of the ith wedge, in left to right order, aims at $f(\beta_i)$, $1 \le i \le n$. Let t_i denote the Backw hit point on L of the ith wedge.

Finally, we connect $s := (-1, 0)$ with $(-1, -1)$ and $(1, -1)$ with $g := (1, 0)$ and with the wedges. This chain is the right side R of P.

The resulting polygon P is walkable. As the guard on R walks from s to g, the guard on L first has to walk to t_1. Then it must turn and backtrack to s, turn and walk to t_2, turn again and backtrack to s, and so forth. To compute the turnpoints of this walk means to compute the edges of L containing the points t_i, $1 \le i \le n$. But evidently, $t_i \in \overline{p_{j-1}p_j} \iff \alpha_{j-1} < \beta_i < \alpha_j$.

Therefore, we can solve our original problem by computing turnpoints. □

7. Conclusions

It is not hard to see how to solve in time $O(n \log n + k)$ the *general counter-walk problem*, where the guards move in opposite directions, with backtracking allowed.

There are some other related questions which may be interesting for further research. Given a polygon P, do there exist two vertices s and g such that P is (straight) walkable? Report all such pairs. In which walk is the maximum distance between the two guards minimized? What if the two guards have to find out *on-line* if the polygon admits a walk, based only on their local visibility information?

Finally, how fast can one decide if a polygon is straight walkable? Our lower bound only applies to computing a general walk of minimum length.

Acknowledgements

We would like to thank the anonymous referees for their valuable comments and suggestions for improvements.

References

1. V. Chvatal, "A Combinatorial Theorem in Plane Geometry", *Journal of Combinatorial Theory B* **13**(6) (1975) 39–41.
2. S. Fisk, "A Short Proof of Chvatals's Watchman Theorem", *Journal of Combinatorial Theory B* **24** (1978) 374.
3. D. T. Lee and A. K. Lin, "Computational Complexity of Art Gallery Problems", *IEEE Transactions on Information Theory* **32** (1986) 276–282.
4. W.-P. Chin and S. Ntafos, "Optimum Watchman Routes", *Information Processing Letters* (1988).
5. J. O'Rourke, *Art Gallery Theorems and Algorithms*, (Oxford University Press, New York, Oxford, 1987).
6. B. Chazelle and L. J. Guibas, "Visibility and Intersection Problems in Plane Geometry", *Discrete and Computational Geometry* **4**(6) (1989) 551–581.
7. L. J. Guibas and J. Hershberger, "Optimal Shortest Path Queries in a Simple Polygon", *Journal of Computer and System Sciences* **39**(2) (1989) 126–152.
8. D. Avis and G. T. Toussaint, "An Optimal Algorithm for Determining the Visibility of a Polygon from an Edge", *IEEE Transactions on Computers* **30** (1981) 910–914.
9. F. P. Preparata and M. I. Shamos, *Computational Geometry*, (Springer-Verlag, New York, 1985).
10. M. Formann and F. Wagner, unpublished note, Freie Universität Berlin, 1990.