Annales Univ. Sci. Budapest., Sect. Comp. 21 (2004) 135-151

VERTEXLIGHTS WITH FIXED DIRECTIONS IN SIMPLE POLYGONS

A. Spillner and H.-D. Hecker
(Jena, Germany)

Abstract. We study the problem of determining the smallest $\alpha \in [0, 2\pi]$ for a given simple polygon P with n vertices, such that P can be illuminated by α-vertexlights the directions of which are fixed. We present an algorithm that finds a solution to this problem in $O(rn^3)$ time, where r is the number of reflex vertices of P. Furthermore we show that with the help of *parametric search* the problem can be solved in $O(rn^2 \log^2 n)$ time. We use the *extended real RAM* as the model of computation.

1. Introduction

We suppose the reader to be familiar with the concepts of polygons and simple polygons. Given a simple polygon P we denote the interior with $int(P)$, the exterior with $ext(P)$ and the boundary with $bd(P)$. The set of vertices of P we denote by $vert(P)$ and the set of edges of P we denote by $edge(P)$. Given a point $x \in P$ we will say that a point $y \in P$ is visible from x, if the segment $\overline{xy}$ is completely contained in P. With $vis(x, p)$ we denote the set of all points in P visible from x. A floodlight is a light source, that projects light inside a cone C. We denote the right bounding ray of C with $right(C)$ and the left bounding ray with $left(C)$. If the size of C is α, we will call C an α-floodlight. If the apex of C is located at a vertex of a polygon, we will call C a vertexlight. We do not allow two floodlights to share a common apex. For convenience we identify a floodlight with the cone it projects light in. Given a simple polygon P and a floodlight F with apex $a \in P$ we will say that a point $y \in P$ is illuminated by F, if y is visible from a and $y \in F$. With $ill(F, P)$ we denote the set of all points in P illuminated by F. Finally we will denote the Euclidean distance between two points x and y by $\|x - y\|$ and the open disk with center c and radius ϱ by $ball(c, \varrho)$.

In [6] it is shown, that for every $\alpha \in [0, \pi)$ there exists a simple polygon, which cannot be illuminated by α-vertexlights. On the other hand every simple polygon with n vertices can be illuminated by at most $n - 2$ π-vertexlights. In [15] it is shown, that it is NP-hard to determine the smallest $\alpha \in [0, \pi]$ such that a given simple polygon can be illuminated by α-vertexlights. We do not know whether or not the corresponding decision problem belongs to the class NP, but one could apply the techniques used in [12] to show that it is at least decidable under reasonable assumptions about the input.

The proof of NP-hardness in [15] relies heavily on the fact that a vertexlight can be rotated around its apex. In this report we try to find out what happens, when the direction of every floodlight used is fixed. We will first study the corresponding decision problem. To solve this we only need to combine well known techniques in computational geometry such as *ray shooting* and the computation of *arrangements of line segments.* Then we use this decision algorithm to construct two algorithms for the minimization problem. The first and less efficient one, with respect to its time complexity, is a rather direct exploitation of the geometry in the problem. The second algorithm is more efficient regarding the running time but only of little practical value since it is based on the *parametric search* technique. A discussion of the application of this technique in computational geometry, its shortcomings and possible alternatives can be found in [3]. For a summary of results in the area of visibility and illumination problem we refer the reader to [17] and [16].

2. The decision problem

First we state the problem in a more formal way:

• `Instance:` A simple polygon P with n vertices and for every vertex v of P a floodlight $F(v)$ with apex v.

`Question:` Is $P = \bigcup_{v \in vert(P)} ill(F(v), P)$?

We will refer to it as problem $\prod_1$.

2.1. Algorithmic tools

Computation of the visibility polygon

For a point $q \in P$ the set $vis(q, P)$ is a simple polygon, which can be determined in $O(n)$ time [13]. From $vis(q, P)$ we can simply extract in $O(n)$ time the vertices of P visible from q.

Ray shooting in simple polygons

In a *ray shooting* query for a given point $g \in P$ and a ray R with starting point q the following should be returned:

- The point q if there is an $\varepsilon > 0$ such that $R \cap ball(q, \varepsilon) \cap int(P) = \emptyset$.
- The point $t \in R$ with $t \neq q$ and $t \in bd(P)$ and $\|q - t\|$ minimal else.

In [10] a procedure is given, with the help of which we can answer such *ray shooting* queries in $O(\log n)$ time after a preprocessing of the polygon P that takes $O(n)$ time.

Arrangements of line segments

A set of line segments will be called simple, if the intersection of every two distinct elements is either a point or empty. Let $\mathfrak{S}$ be a simple set of line segments. We are interested in the following planar graph $G(\mathfrak{S})$:

- The set of vertices $\mathfrak{V}$ contains every endpoint of a segment in $\mathfrak{S}$ and every point that is the intersection of two distinct segments.
- Two distinct vertices u and v from $\mathfrak{V}$ will be connected by an edge, if there is a segment $S \in \mathfrak{S}$ such that $\overline{uv} \subseteq S$ and there is no vertex w distinct from u and v with $w \in \overline{uv}$.

In [5] an algorithm is given, which computes the trapezoidal decomposition $T(\mathfrak{S})$ of $\mathfrak{S}$ as *DECL (Doubly Connected Edge List*, [14]). Additionally we have for every edge E of $T(\mathfrak{S})$ a pointer to the segment in $\mathfrak{S}$ containing E. The algorithm runs in $O(|\mathfrak{S}| \log |\mathfrak{S}| + |\mathfrak{V}|)$ time and handels all possible degenerate cases explicitely. But from $T(\mathfrak{S})$ we can easily obtain $G(\mathfrak{S})$ in $O(|\mathfrak{V}|)$ time by deleting appropriate edges.

A lexicographic order on line segments

For points in the plane we have the usual lexicographic order. We introduce a similar order for line segments. Let $S_1 = \overline{p_1 q_1}$ and $S_2 = \overline{p_2 q_2}$ be line segments. W.l.o.g. we suppose that $p_1 \leq q_1$ and $p_2 \leq q_2$, otherwise we would rename the points. We will say $S_1 \leq S_2$ if either $p_1 < p_2$ or $p_1 = p_2$ and $q_1 \leq q_2$.

2.2. The algorithm

Our algorithm uses the visibility cell decomposition of a simple polygon which can be found in [8] and [11].

Algorithm 2.1

(1) Determine a list L_{ref} of the reflex vertices of P.

(2) Determine for every $v \in L_{ref}$ a list $L_{vis}(v)$ of all those vertices of P visible from v and distinct from v.

(3) Remove for every $v \in L_{ref}$ from the List $L_{vis}(v)$ all those elements w with $v \notin ill(F(w), P)$.

(4) Initialize an empty balanced search tree $\mathcal{T}$ for line segments. Every segment $S = \overline{p_1 p_2}$ stored in $\mathcal{T}$ will be part of the boundary of $ill(F(v), P)$ for at least one $v \in vert(P)$. For S we maintain an entry $bor(p_1, p_2)$ which is the number of floodlights F, such that $int(ill(F, P))$ is to the right when we traverse S from p_1 to p_2. Similarly we maintain an entry $bor(p_2, p_1)$.

(5) Perform for every $v \in vert(P)$ and the ray $left(F(v))$ *a ray shooting* query in P. Let t be the point returned. If $t \neq v$, we will search the segment $\overline{vt}$ in $\mathcal{T}$. If $\overline{vt}$ is not found, we will insert it, set $bor(v, t) = 1$ and set $bor(t, v) = 0$. If we find $\overline{vt}$ in $\mathcal{T}$, we will increase $bor(v, t)$ by one. We do the same for every $v \in vert(P)$ and the ray $right(F(v))$.

(6) Perform for every $v \in L_{ref}$ and every $w \in L_{vis}(v)$ a *ray shooting* query with v as starting point and with the ray which is contained in the ray $\vec{wv}$. Let t be the point returned. If $v \neq t$, we will proceed analogously to Step (5).

(7) Compute the graph $G(\mathfrak{S})$ as defined above, where $\mathfrak{S}$ is the set of segments in $\mathcal{T}$ together with the edges of P.

(8) Choose a face C in $G(\mathfrak{S})$ and determine a point $q \in int(C)$. Determine the number of floodlights F with $q \in ill(F, P)$. Start at C a breadth-first search on the faces of $G(\mathfrak{S})$. When we cross an edge E from a face C_1 to a face C_2 we determine the segment $S = \overline{p_1 p_2}$ containing E. With the help of $bor(p_1, p_2)$ and $bor(p_2, p_1)$ we can from the number of floodlights illuminating C_1 determine the number of floodlights illuminating C_2. If every face of $G(\mathfrak{S})$ is illuminated by at least one floodlight we output `yes`, else `no`.

Theorem 2.1. *Algorithm 2.1 works correctly and it runs in $O(rn \log n + k)$ time with $k \in O(rn^2)$, where r is the number of reflex vertices of P and k is the number of vertices of the constructed planar graph. Furthermore we need $O(k)$ space to store this graph in the DECL.*

Proof. The algorithm computes a planar graph which is related to the visibility cell decomposition of P. It is easy to see that the interior of a face of this graph is either completely contained in the region illuminated by a floodlight or it is disjoint to it. In determining for every face the number of floodlights illuminating it, we can decide whether or not P is completely illuminated.

The proof that the constructed graph has $O(rn^2)$ vertices can be done as in [11]. Here we only add the $2n$ bounding rays of the floodlights.

So let us turn to the analysis of the time complexity of our algorithm. Step (1) can be done in $O(n)$ time by a single traversal of the vertices of P. Step (2) can be done in $O(rn)$ time by determining the visibility polygon for every reflex vertex of P. Step (3) can also be done in $O(rn)$ time, since we only have to test whether or not $v \in F(w)$. In Step (5) after the above mentioned preprocessing of P a *ray shooting* query and an update of the balanced search tree can be done in $O(\log n)$ time, which gives altogether $O(n \log n)$ time. Similarly Step (6) takes $O(rn \log n)$ time. It is not hard to see, that $\mathfrak{S}$ is a simple set of line segments. In Step (7) we use the above mentioned algorithm from [5]. Since we have $O(rn)$ segments, this takes us $O(rn \log n + k)$ time. The breadth-first search on the graph can be done in $O(k)$ time with the help of the information stored in the *DECL*.

Corollary 2.1. *When P is a convex polygon, the problem $\prod_1$ can be decided in $O(n \log n + k)$ time with $k \in O(n^2)$.*

3. Minimizing the size of the given floodlights

For a point p, a ray R with starting point p and a $\beta \in [0, 2\pi]$ we denote by $F(p, R, \beta)$ the floodlight with apex p, angular bisector R and size β. Occasionally we will refer to R as the direction of the floodlight. With this we state the following minimization problem $\prod_2$:

- `Instance:` A simple program P with n vertices and for every vertex v a ray $R(v)$ with starting point v.

  `Objective:` Determine α, which is the smallest $\beta \in [0, 2\pi]$ such that $P = \\ = \bigcup_{v \in vert(P)} ill(F(v, R(v), \beta), P)$?

3.1. Restricting the problem

As a first step we will attack the following problem $\prod_3$:

- `Instance:` A convex polygon Q, a finite nonempty set M of points and for every point p in M a ray $R(p)$ with starting point p and $int(Q) \cap R(p) = \emptyset$.

`Objective:` Determine α, which is the smallest $\beta \in [0, 2\pi]$ such that $Q \subseteq \subseteq \sum_{p \in M} F(p, R(p), \beta)$?

For every $\beta \in [0, 2\pi]$ we set

$$\mathfrak{B}(M, \beta) = \{left(F(p, R(p), \beta)) : p \in M\} \bigcup \{right(F(p, R(p), \beta)) : p \in M\}.$$

Lemma 3.1. *For every $\beta \in [0, 2\pi]$ the set $cl(Q \setminus \bigcup_{p \in M} F(p, R(p), \beta))$ is empty or a convex polygon.*

Proof. Since for every $p \in M$ we have $int(Q) \cap R(p) = \emptyset$, it is not hard to see that $Q \setminus \bigcup_{p \in M} F(p, R(p), \beta)$ is the intersection of the closed halfplanes determined by the straight lines containing an edge of Q and some of the open halfplanes determined by the straight lines containing a bounding ray of a floodlight.

Lemma 3.2. *It is $\alpha = 2\pi$ or there exist mutual distinct elements E_1, E_2 and E_3 from $\mathfrak{B}(M, \alpha) \bigcup edge(Q)$ and a point p with $p \in E_1 \cap E_2 \cap E_3$.*

Proof.

Case 1. ($\alpha = 2\pi$). Then we are done.

Case 2. ($\alpha < 2\pi$). There could be a $q \in M$ with $Q \subseteq \bigcup_{r \in M \setminus \{q\}} F(r, R(r), \alpha)$. Then we consider the set $M^* = M \setminus \{q\}$, because $\mathfrak{B}(M^*, \alpha) \subseteq \mathfrak{B}(M, \alpha)$ and $min\{\beta \in [0, 2\pi] : Q \subseteq \bigcup_{r \in M^*} F(r, R(r), \beta)\} = \alpha$. So w.l.o.g. there is a $q \in M$ with $Q \not\subseteq \bigcup_{r \in M \setminus \{q\}} F(r, R(r), \alpha)$. Then we know that $C = = cl(Q \setminus \bigcup_{r \in M \setminus \{q\}} F(r, R(r), \alpha))$ is a convex polygon and $C \subseteq F(q, R(q), \alpha)$.

Subcase 2.1 ($q \notin C$). Then there is a $v \in vert(C)$ with

$$v \in left(F(q, R(q), \alpha)) \bigcup right(F(q, R(q), \alpha))$$

since otherwise we would obtain a contradiction to the minimality of α. However v is the point of intersection between two distinct elements of $\mathfrak{B}(M \setminus \{q\}, \alpha) \bigcup edge(Q)$.

Subcase 2.2 ($q \in C$). Then it is $q \in bd(C) \cap bd(Q)$.

Subsubcase 2.2.1 ($q \notin vert(C)$). Then q lies in the relative interior of an edge E of C. But then at least one of the endpoints of E is an element of $left(F(q, R(q), \alpha)) \bigcup right(F(q, R(q), \alpha))$, since otherwise we would obtain a contradiction to the minimality of α. This endpoint of E again is the intersection between two distinct elements of $\mathfrak{B}(M\backslash\{q\},\alpha)\bigcup edge(Q)$.

Subsubcase 2.2.2 ($q \in vert(C)$). Then at least two distinct elements E_1 and E_2 of $\mathfrak{B}(M\backslash\{q\},\alpha)\bigcup edge(Q)$ intersect at the point q, such that two edges of C incident to q are contained in E_1 and E_2 respectively. If E_1 and E_2 are edges of Q, q will be a vertex of Q and because of the minimality of α there is a vertex v of Q distinct from q such that $v \in left(F(q,R(q),\alpha))\bigcup right(F(q,R(q),\alpha))$. So at the point v intersect two edges of Q and a bounding ray of a floodlight the apex of which is distinct from v. If E_1 is an edge of Q and E_2 is a bounding ray of a floodlight, then at q two distinct bounding rays and an edge of Q intersect. If E_1 and E_2 are bounding rays, then they belong to distinct floodlights and so three mutual distinct bounding rays intersect at the point q.

So a strategy for finding α could be to determine the set $\mathfrak{C}$ of all those $\beta \in [0,2\pi]$ where three mutual distinct elements from $\mathfrak{B}(M,\beta)\bigcup edge(Q)$ have nonemty intersection. Then we could search for the smallest $\beta \in \mathfrak{C}$ with $Q \subseteq \bigcup\limits_{p\in M} F(p,R(p),\beta)$. However at the moment it is not clear whether or not searching in $\mathfrak{C}$ will be easier than searching in the whole interval $[0,2\pi]$ for α. We first have to explore some properties of $\mathfrak{C}$.

From now on we will not consider the bounding rays of the floodlights or the edges of Q itself, but the straight lines containing them. In this sense we will occasionally speak about the straight line corresponding to a bounding ray or an edge of Q. Doing so, we will not miss an element of $\mathfrak{C}$ since when three elements of $\mathfrak{B}(M,\beta)\bigcup edge(Q)$ for any $\beta \in [0,2\pi]$ have nonempty intersection the three corresponding straight lines also have. On the other hand we gain more clarity in the following argumentations.

Next we introduce an appropriate parameter for the straight lines corresponding to the bounding rays in $\mathfrak{B}(M,\beta)$ when $\beta \in \left[0,\frac{\pi}{2}\right]$, since the use of β itself caused some trouble. The ranges $\beta \in \left(\frac{\pi}{2},\frac{3\pi}{2}\right]$ and $\beta \in \left(\frac{3\pi}{2},2\pi\right]$ are treated analogously. For every $p \in M$ we suppose that the direction of the ray $R(p)$ is given by the vector $(a(p),b(p))$. Please have a look at Figure 1. Then the direction of $right(F(p,R(p),\beta))$ is given by the vector $(a(p),b(p))+\mu(-b(p),a(p))$ and the direction of $left(F(p,R(p),\beta))$ is given by the vector $(a(p),b(p))+\mu(b(p),-a(p))$ with $\mu = tan\left(\frac{\beta}{2}\right)$. So μ varies within the interval $[0,1]$.

Now suppose $Ax + By + C = 0$ is the equation of the straight line corresponding to $left(F(p,R(p),\beta))$. Then it is not hard to see that A, B and C are linear expressions in μ. The same is true for the equation of the straight line corresponding to $right(F(p, R(p), \beta))$. For the equation of a straight line corresponding to an edge of Q, these coefficients are even constant. Now we examine the intersection of every three distinct straight lines G_1, G_2 and G_3. Let $A_i x + B_i y + C_i = 0$ be the equation of G_i, $i \in \{1,2,3\}$. When $G_1 \cap G_2 \cap G_3 \neq \emptyset$, we have

$$det \begin{pmatrix} A_1 & B_1 & C_1 \\ A_2 & B_2 & C_2 \\ A_3 & B_3 & C_3 \end{pmatrix} = 0.$$

This determinant is a polynomial $H(\mu)$ in μ with degree at most three. Now there can occur two cases:

Case 1 ($H(\mu) \not\equiv 0$). Then $H(\mu)$ has at most three distinct roots. So there are at most three interesting values of μ for G_1, G_2 and G_3.

Case 2 ($H(\mu) \equiv 0$). It will turn out that those tripel of straight lines need not be considered. However just to see this there is still some work to do.

Consider a point $p \in M$ and the floodlight $F(p, R(p), \beta)$. When β increases from 0 to 2π the ray $right(F(p, R(p), \beta))$ performs a clockwise rotation around its starting point p. Similarly $left(F(p, R(p), \beta))$ performs a counterclockwise rotation around p. Since we deal with the straight lines containing the bounding rays, we are interested in the behavior of the intersection of rotating straight lines. In what follows we will denote by $rot_{c,\delta}$ the map of the plane on itself which consists in a clockwise rotation around the point c by an angle of size δ.

Lemma 3.3. *Let L_1 and L_2 be distinct straight lines and $c_1 \in L_1$ and $c_1 \in L_2$ with $c_1 \neq c_2$. We let L_1 rotate in clockwise direction around c_1 and L_2 in clockwise direction around c_2 and consider the set*

$$C = \{p \in rot_{c_1,\beta}(L_1) \cap rot_{c_2,\beta}(L_2) : \beta \in [0, 2\pi]\}.$$

Then C is the straight line through c_1 and c_2 or a circle containing c_1 and c_2.

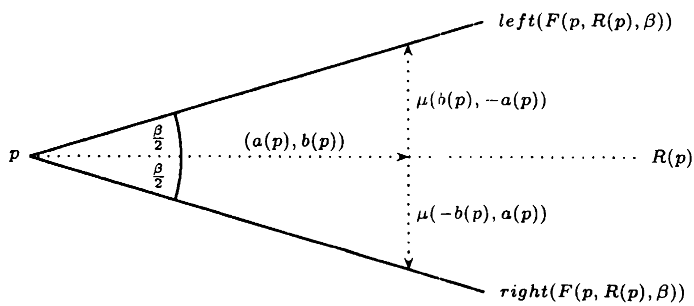

Figure 1. Introduction of the parameter μ

Proof. Since $c_1 \neq c_2$ there are only two possible cases:

Case 1 (L_1 is parallel to L_2). Then the images of L_1 and L_2 during the rotation remain parallel. So they can only have nonempty intersection, when they both coincide with the straight line through c_1 and c_2.

Case 2 (L_1 and L_2 intersect in a point q). Then the intersection of the images of L_1 and L_2 during the rotation is always a single point, namely the image of q and its not hard to see that C is a circle through c_1 and c_2.

Lemma 3.4. *Let L_1 and L_2 be distinct straight lines and $c_1 \in L_1$ and $c_2 \in L_2$ with $c_1 \neq c_2$. We let L_1 rotate in clockwise direction around c_1 and L_2 in counterclockwise direction around c_2 and consider the set*

$$H = \{p \in rot_{c_1,\beta}(L_1) \cap rot_{c_2,2\pi-\beta}(L_2) : \beta \in [0, 2\pi]\}.$$

Then H is a possibly degenerate hyperbola.

Proof. There are exactly two values for β in $[0, 2\pi]$ such that $rot_{c_1,\beta}(L_1)$ and $rot_{c_2,2\pi-\beta}(L_2)$ are parallel. This gives us the direction of the asymptotes of the hyperbola. To verify that H is indeed a hyperbola we can show, that the points in H are the solution of an appropriate equation with degree two. The details will not be given here, since actually we are only interested in the qualitative structure of H.

We will now return to the study of those triples G_1, G_2, G_3 of straight lines corresponding to bounding rays or edges of Q, where the polynomial $H(\mu) \equiv 0$. We will distinguish four possible subcases:

Subcase 2.1 (G_1, G_2 and G_3 correspond to distinct edges of Q). Since Q is a convex polygon, it is not hard to see, that $G_1 \cap G_2 \cap G_3 = \emptyset$, which means we do not need to consider such triples.

Subcase 2.2 (G_1 and G_2 correspond to distinct edges of Q and G_3 corresponds to a bounding ray R). It might be that G_1 and G_2 are parallel. Then we do not need to consider the tripel. So we suppose that G_1 and G_2 intersect in a point q. Since q is fixed $H(\mu) \equiv 0$ is only possible when q is the starting point of R. However by the proof of Lemma 3.2 we know, that we do not need to consider such tripels.

Subcase 2.3 (G_1 corresponds to an edge of Q and G_2 and G_3 correspond to distinct bounding rays R_2 and R_3 respectively). We first suppose that R_2 and R_3 belong to the same floodlight. Then they intersect in the apex of this floodlight which is not interesting and we need not consider such tripels. So we now suppose that R_2 and R_3 belong to distinct floodlights. Then G_2 and G_3 rotate around distinct points.

Subsubcase 2.3.1 (G_2 and G_3 rotate in the same direction). It could be that G_2 and G_3 are parallel, but then $H(\mu) \equiv 0$ would mean, that G_1 and G_2 must be parallel, too, which is impossible since G_1 is fixed. So G_2 and G_3

intersect in a single point, which by Lemma 3.3 describes a circle during the rotation. On the other hand $H(\mu) \equiv 0$ means that the point of intersection between G_2 and G_3 is always on the fixed straight line G_1, which is impossible.

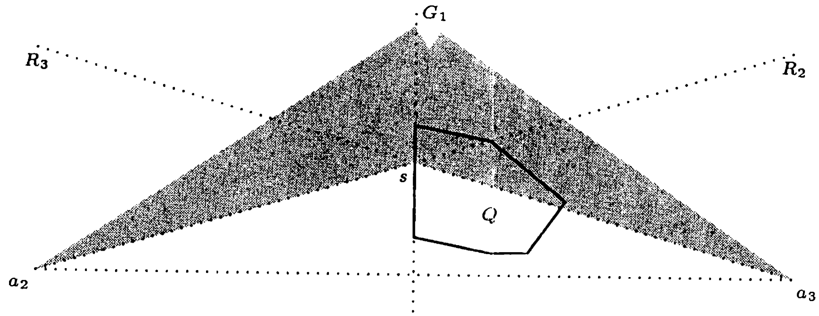

Figure 2. The point of intersection s between R_2 and R_3 varies on G_1

Subsubcase 2.3.2 (G_2 and G_3 rotate in opposite directions). Then by Lemma 3.4 the point of intersection s between G_2 and G_3 describes a hyperbola. On the other hand this point must always be on G_1, since $H(\mu) \equiv 0$. This is only possible when the hyperbola is degenerate and G_1 is the bisector of the segment $\overline{a_2a_3}$, where a_2 is the starting point of R_2 and a_3 is the starting point of R_3. Please have a look at Figure 2. W.l.o.g. we suppose the polygon Q to be to the right of G_1. It is not hard to see that only those sizes of the floodlights are interesting where an additional straight line G_4 goes through the point s. G_4 can correspond to an edge of Q. We find the interesting sizes for the floodlights, when we consider the tripel G_1, G_2 and G_4. On the other hand G_4 can correspond to a bounding ray R_4. Then we suppose w.l.o.g. that G_3 and G_4 rotate in the same direction and we find the interesting sizes for the floodlights when we consider the tripel G_1, G_3, G_4. Again it turns out that we need not consider the tripel G_1, G_2, G_3.

Subcase 2.4 (G_1, G_2 and G_3 correspond to distinct bounding rays R_1, R_2 and R_3 respectively). When two of the bounding rays have the same starting point, then it is impossible that $H(\mu) \equiv 0$. Thus we suppose that the starting points a_1, a_2 and a_3 of R_1, R_2 and R_3 respectively are mutually distinct.

Subsubcase 2.4.1 (G_1, G_2 and G_3 rotate in the same direction). Then $H(\mu) \equiv 0$ either means that G_1, G_2 and G_3 are parallel or by Lemma 3.3 G_1, G_2 and G_3 intersect in a single point s which describes the circle through a_1, a_2 and a_3. From the proof of Lemma 3.2 we know, that we need not consider a tripel of parallel straight lines. So please have a look at Figure 3. It is not hard to see that only those sizes of the floodlights are interesting where an additional straight line G_4 goes through the point s. G_4 can correspond to an edge of Q. We find the interesting sizes for the floodlights, when we consider the tripel G_1, G_2 and G_4. On the other hand G_4 can correspond to a bounding ray R_4. It is not hard to see that only those straight lines G_4 are interesting, where G_1 and G_4 do not intersect in a single point which describes the circle through a_1, a_2 and a_3. We find the interesting sizes for the floodlights when we consider the tripel G_1, G_2 and G_4 and we need not consider the tripel G_1, G_2 and G_3.

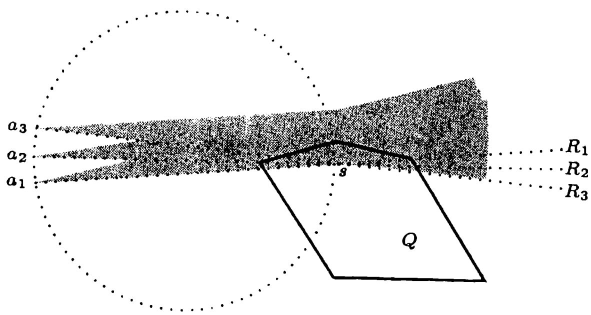

Figure 3. The point of intersection s between R_1, R_2 and R_3 describes a circle

Subsubcase 2.4.2 (G_1, G_2 and G_3 do not all rotate in the same direction). W.l.o.g. we suppose that G_1 and G_2 rotate in the same direction. Then by Lemma 3.4 the point of intersection between G_2 and G_3 describes a hyperbola and by Lemma 3.3 either G_1 and G_2 are parallel or their point of intersection describes a circle. It is not hard to see that $H(\mu) \equiv 0$ is impossible.

Since we have exhausted all possibilities we have established that all the tripels of straight lines with $H(\mu) \equiv 0$ are not relevant for finding the interesting sizes of the floodlights. So we are ready to give an algorithm for finding the minimal size of the floodlights such that the polygon Q is contained in them.

Algorithm 3.1

(1) Determine for every edge E of Q the straight line G containing E, i.e. the coefficients of the equation of G. Store G in a list L_{edge}.

(2) Determine for every bounding ray R the straight line G containing R, i.e. the coefficients of the equation of G. Store G in a list L_{ray}.

(3) For every vertex v of Q and every straight line G in L_{ray} determine the relevant values for μ where $v \in G$ and insert those values in a list L_{rel}.

(4) For every straight line G_1 in L_{edge} and every two distinct straight lines G_2 and G_3 in L_{ray} determine the polynomial $H(\mu)$. If $H(\mu) \not\equiv 0$ then determine the real roots of $H(\mu)$ and insert them in the list L_{rel}.

(5) For every three distinct straight lines G_1, G_2 and G_3 in L_{ray} determine the polynomial $H(\mu)$. If $H(\mu) \not\equiv 0$ then determine the real roots of $H(\mu)$ and insert them in the list L_{rel}.

(6) While L_{rel} contains at least two distinct values do:

(6.1) Determine the median μ_{med} of the values in L_{rel} and the lists

$$L_{less} = \{\mu \in L_{rel} : \mu_{rel} \leq \mu_{med}\}, \ L_{great} = \{\mu \in L_{rel} : \mu > \mu_{med}\}.$$

(6.2) Test with algorithm 3.2 if Q is contained in the union of the floodlights for μ_{med}. If this is true then set $L_{rel} = L_{less}$ else set $L_{rel} = L_{great}$.

(7) Output an element of L_{rel} as a value for the parameter where the size of the floodlights is minimal.

Algorithm 3.2. We want to test, whether or not Q is contained in the union of the floodlights when the size of them is β.

(1) The set $cl\left(Q \setminus \bigcup_{p\in M} F(p, R(p), \beta)\right)$ is contained in the intersection of the halfspaces determined by the edges of Q and some halfspaces determined by the bounding rays of the floodlights (Lemma 3.1). Compute a list L of these halfspaces.

(2) Compute the intersection of the halfspaces in L.

(3) If the interior of this intersection is empty output `yes` else `no`.

Theorem 3.1. *With Algorithm 3.1 we can find a value for the parameter, such that the floodlights have minimal size α, in $O\left(m^3 + nm^2\right)$ time, where n is the number of vertices of Q and m is the number of points in M. We use the extended real RAM as the model of computation. In fact we suppose that we can compute square roots and cubic roots of real numbers in $O(1)$ time. Furthermore we need $O(m^3 + nm^2)$ space for the list L_{rel}.*

Proof. That our algorithm solves the problem, is an immediate consequence of the considerations preceding the algorithm. So we can turn our attention to the analysis of the time complexity. Step (1) takes $O(n)$ and Step (2) $O(m)$ time. For Step (3) we need $O(nm)$ time. Step (4) can be done in $O(nm^2)$ and Step (5) in $O(m^3)$ time. The loop in Step (6) is passed at most $O(\log(m^3 + nm^2))$ time.

To determine the running time of one pass of this loop we need to know the time complexity of Algorithm 3.2. For one $p \in M$ we can determine the points of intersection between the bounding rays of the floodlight and the boundary of Q in $O(\log n)$ time. Thus we can compute the list L in $O(m \log n)$ time. We know that L contains $O(n+m)$ halfspaces the intersection of which can be computed in $O((n+m)\log(n+m))$ time. Summing up over all passes of the loop in Step (6) gives $O((m+n)\log^2(m+n))$.

The time for determining the median and the corresponding lists over all passes of the loop gives $O(m^3+nm^2)$, which is thus the time complexity of the whole algorithm.

3.2. A first algorithm for problem $\prod_2$

We now want to use the methods developed in the previous subsection to solve the minimization problem $\prod_2$. Therefore we will decompose P in suitable convex polygons. This is done in the first part of the following algorithm. This decomposition again is related to the visibility cell decomposition of a simple polygon [8], [11].

Algorithm 3.3

(1) Determine a list L_{ref} of the reflex vertices of P.

(2) Determine for every $v \in L_{ref}$ the list $L_{vis}(v)$ of all those vertices of P which are visible from v and distinct from v.

(3) Initialize an empty balanced search tree $\mathcal{T}$ for line segments.

(4) Perform for every $v \in vert(P)$ and the ray $R(v)$ a *ray shooting* query in P. Let t be the point returned. If $t \neq v$, then try to insert the segment $\overline{vt}$ in $\mathcal{T}$.

(5) Perform for every $v \in L_{ref}$ and every $w \in L_{vis}(v)$ a *ray shooting* query with starting point v and the ray which is contained in the ray $\vec{wv}$. Let t be the point returned. If $t \neq v$, then try to insert $\overline{vt}$ in $\mathcal{T}$.

(6) Compute the planar graph $G(\mathfrak{S})$ as in Algorithm 2.1, where $\mathfrak{S}$ is the set of line segments in $\mathcal{T}$ together with $edge(P)$.

(7) Determine for every segment E in $\mathfrak{S}$ the straight line G containing E. Store G in a list L_{edge}.

(8) Determine for every bounding ray R the straight line G containing R, i.e. again the coefficients of the equation of G as linear functions of the parameter μ. Store G in a list L_{ray}.

(9) For every vertex v of $G(\mathfrak{S})$ and every straight line G in L_{ray} determine the relevant values for the parameter μ where $v \in G$. Store those values in a list L_{rel}.

(10) For every straight line G_1 in L_{edge} and every two distinct straight lines in L_{ray} determine the polynomial $H(\mu)$. If $H(\mu) \not\equiv 0$ then find the real roots of $H(\mu)$ and insert them in L_{rel}.

(11) For every three straight lines G_1, G_2 and G_3 in L_{ray} determine the polynomial $H(\mu)$. If $H(\mu) \not\equiv 0$ then find the real roots of $H(\mu)$ and insert them in L_{rel}.

(12) While L_{rel} contains at least two distinct values do:

(12.1) Determine the median μ_{med} of the values in L_{rel} and the lists

$$L_{less} = \{\mu \in L_{rel} : \mu \leq \mu_{med}\}, \ L_{great} = \{\mu \in L_{rel} : \mu > \mu_{med}\}.$$

(12.2) Test with Algorithm 2.1 whether or not for the value μ_{med} the polygon P is completely illuminated by the floodlights. If this is true then set $L_{rel} = L_{less}$ else set $L_{rel} = L_{great}$.

(13) Output an element of L_{rel} as a value of the parameter μ where the size of the floodlights is minimal.

Theorem 3.2. *With Algorithm 3.3 we can determine a value for the parameter such that the floodlights have minimal size in $O(rn^3)$ time, where r is the number of reflex vertices of P. We use the extended real RAM as the model of computation. Furthermore we need $O(rn^3)$ space for the list L_{rel}.*

Proof. It is not hard to see that the faces of the planar graph $G(\mathfrak{S})$ are convex polygons and by construction of $G(\mathfrak{S})$ the rays which determine the directions of the floodlights do not intersect the interior of any face. So the interesting values for the parameter are collected in L_{rel}. In fact L_{rel} will possibly contain a lot more elements than necessary, since for a face of the graph we do not distinguish between vertices which can see this face and those which can not. However from the values in L_{rel} we can determine the smallest that suffices for illumination of P by the floodlights.

The analysis of the running time is a combination of the analysis of Algorithm 3.1 and Algorithm 2.1, so we do not need to go into details. Please note that $G(\mathfrak{S})$ again has $O(rn^2)$ vertices and that L_{rel} contains $O(rn^3)$ elements.

Corollary 3.1. *If P is a convex polygon Algorithm 3.3 runs in $O(n^3)$ time.*

3.3. A second algorithm for problem $\prod_2$

We now want to use the *parametric search* technique. It can be applied to the problem $\prod_2$ as follows. If we have a sequential algorithm A_s running in T_s time and a parallel algorithm A_p running in T_p time using N_p processors for the decision problem $\prod_1$, then we can construct a sequential algorithm for the problem $\prod_2$ running in $O(T_p N_p + T_s T_p \log N_p)$ time.

We already have a sequential algorithm A_s, namely Algorithm 2.1. Thus we have $T_s \in O(rn^2)$. What remains to do is describing a parallel algorithm for the problem $\prod_1$.

Lemma 3.5. *Problem $\prod_1$ can be solved by a parallel algorithm in $O(\log n)$ time using $O(rn^2)$ processors on a CRCW-PRAM.*

Proof. We will show that every step of Algorithm 2.1 can be executed in $O(\log n)$ time with $O(rn^2)$ processors on a CRCW-PRAM. For Step (1) this is clear. In Step (2) for the computation of the visibility polygon of a reflex vertex we use the algorithm presented in [2]. It runs in $O(\log n)$ time and uses $O\left(\frac{n}{\log n}\right)$ processors. So for computing the visibility polygon for all r reflex vertices parallel by we need $O(rn)$ processors. Parallelization of Step (3) is easy. In [10] it is shown that the preprocessing of the polygon P for the *ray shooting* queries in Steps (5) and (6) can be done in $O(\log n)$ time with $O\left(\frac{n}{\log n}\right)$ processors. The *ray shooting* queries itself can be performed parallelly each in $O(\log n)$ time. Instead of a tree for storing the line segments we use a list, which is sorted afterwards and duplicate elements are removed. All this can be done in $O(\log n)$ time using $O(rn)$ processors. For computation of the graph G in Step (7) we use the algorithm from [9] which for given l line segments computes the trapezoidal decomposition of these line segments in $O(\log l)$ time using $O(l \log l + m)$ processors where m is the number of intersecting pairs of line segments. Thus we can determine the graph G in $O(\log n)$ time with $O(rn^2)$ processors. Next we compute a spanning tree T of the dual graph of G. This can be done in $O(\log n)$ time using $O(rn^2)$ processors by an algorithm presented in [18]. Then we root T at an arbitrary vertex w and determine the number of floodlights that illuminate the face of G corresponding to w. Using *Euler-Tour* technique and a prefix-sum algorithm we determine for every vertex of T the number of floodlights illuminating the corresponding face of G in $O(\log n)$ time with $O(rn^2)$ processors and decide thus problem $\prod_1$.

Remark 3.1. Similar ideas for developing a parallel decision algorithm were used in [1] for computing the k-th smallest distance of a set of points in the plane and in [4] for computing the minimum Hausdorff distance between polygon objects.

With the sequential and the parallel algorithm together we have the following

Theorem 3.3. *We can solve the problem* $\prod_2$ *in* $O(rn^2 \log^2 n)$ *time with a sequential algorithm using parametric search.*

4. Concluding remarks

We consider the problem of illuminating a simple polygon with vertexlights the size of which is as small as possible. It could be interesting to study similar problems in the world of simple rectilinear polygons. From [7] we know, that for every $\sigma \in \left[0, \frac{\pi}{2}\right)$ there is a simple rectilinear polygon, which cannot be illuminated by σ-vertexlights and every simple rectilinear polygon can be illuminated by $\frac{\pi}{2}$-vertexlights.

References

[1] **Agarwal P.K., Aronov B., Sharir M. and Suri S.,** Selecting distance in the plane, *6th Annual Symposium on Computational Geometry,* 1990, 321-331.

[2] **Atallah M.J., Chen D.Z. and Wagener H.,** An optimal parallel algorithm for the visibility of a simple polygon from a point, *J. Assoc. Comput. Mach.,* **38** (1991), 516-533.

[3] **Agarwal P.K. and Sharir M.,** Efficient algorithms for geometric optimization, *ACM Computing Surveys,* **30** (1998), 412-458.

[4] **Agarwal P.K., Sharir M. and Toledo S.,** Applications of parametric searching in geometric optimization, *ACM-SIAM Symposium on Discrete Algorithms,* 1992, 72-82.

[5] **Burnikel C.,** *Exact computation of Voronoi diagrams and line segment intersection,* PhD thesis, Universität Saarbrücken, 1996.

[6] **Estivill-Castro V., O'Rourke J., Urrutia J. and Xu D.,** Illumination of polygons with vertex lights, *Information Processing Letters,* **56** (1995), 9-13.

[7] **Estivill-Castro V. and Urrutia J.,** Optimal floodlight illumination of orthogonal polygons, *Proc. Canadian Conference on Computational Geometry,* 1994, 81-86.

[8] **Guibas L.J., Motwani R. and Raghavan P.,** The robot localization problem, *SIAM Journal on Computing,* **26** (1997), 1120-1138.

[9] **Goodrich M.T.,** Intersecting line segments in parallel with an output-sensitive number of processors, *SIAM Journal on Computing,* **20** (1991), 737-755.

[10] **Hershberger J. and Suri S.,** A pedestrian approach to ray shooting: Shoot a ray, take a walk, *Proceedings 4th ACM-SIAM Symposium on Discrete Algorithms,* 1993, 54-63.

[11] **Klein R.,** *Algorithmische Geometrie,* Addison Wesley, 1997.

[12] **O'Rourke J.,** The complexity of computing minimum convex covers for polygons, *Proc. 20th Allerton Conference,* 1982, 75-84.

[13] **O'Rourke J.,** *Art gallery theorems and algorithms,* Oxford University Press, 1987, 203-206.

[14] **Preparata F.P. and Shamos M.I.,** *Computational geometry: An introduction,* Springer, 1985.

[15] **Spillner A. and Hecker H.D.,** Minimizing the size of vertexlights in simple polygons, *Mathematical Logic Quaterly* (to appear)

[16] **Shermer T.,** Recent results in art galleries, *Proc. of the IEEE,* (1990), 1384-1399.

[17] *Handbook of computational geometry,* eds. J.R. Sack and J. Urrutia, North Holland, 1998.

[18] **Shiloach Y. and Vishkin U.,** An $O(\log n)$ parallel connectivity algorithm, *J.of Algorithms,* **3** (1982), 57-67.

(Received November 21, 2001)

A. Spillner and H.-D. Hecker
Department of Algorithms
Faculty of Mathematics and Computer Science
Friedrich Schiller University
Ernst Abbe Platz 1-4
D-07740 Jena, Germany