Two-Guarding Simple Polygons

Patrice Belleville (patrice@cs.sfu.ca)[1]
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6

Abstract

A set $\mathcal{P} = P_1, P_2, \ldots, P_k$ of polygons is called a *k-cover* of a simple polygon P if $P = \cup_{i=1}^{k} P_i$. It has been shown that, in most cases, finding k-covers of polygons by subpolygons of a given class is NP-hard. Shermer [She91a] gave a linear time algorithm to determine if P has a two-cover by convex polygons; Belleville [Bel91] gave an $O(n^2)$ time algorithm that finds two-covers by L_k-convex polygons. In this paper, we give an algorithm to find a two-cover of P by star-shaped polygons in $O(n^4)$ time and $O(n^2 \log n)$ space respectively, if one exists.

1 Introduction

This section introduces the notation and the definitions that will be used throughout this paper. If p is a point of the plane, we denote by $N_\varepsilon(p)$ the open circle of radius ε centered at p. The *interior* of a line segment l, denoted by $int(l)$, is the open line segment having the same endpoints as l. Given a simple polygon P, we denote the boundary of P by $bd(P)$, its interior by $int(P)$, and its exterior by $ext(P)$.

A point p of P is *visible* from a point p' of P if the line segment joining p to p' does not intersect $ext(P)$. We say that a subset Q of P is *completely visible* from another subset Q' of P if every point of Q is visible from every point of Q', and denote by $V(P, Q)$ the subset of P from which Q is completely visible. A subset Q of P is said to be *weakly visible* from a subset Q' of P if, for each point q of Q, there exists a point of Q' (depending on q) that is visible from q.

The *upper envelope* of a set $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ of functions over an interval I is the function f, with domain I, defined for $t \in I$ by $f(t) = max\{f_i(t) | f_i \in \mathcal{F}\}$. The lower envelope of $\mathcal{F}$ is defined similarly. Envelopes are discussed further in Sections 2.2 and 3.2.

This paper is mainly concerned with computing *k-covers* of simple polygons. A set $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ is said to be a *k-cover* for a simple polygon P if $P = \cup_{i=1}^{k} P_i$. Results on k-covers of simple polygons can be found in [O'R87, She91b].

[1]This work was carried out while the author was with the School of Computer Science, McGill University, Montréal, Québec, Canada [Bel91].

The *Two-Cover* problem for a given class of polygons consists in determining whether a polygon P can be covered by two polygons of this class. Shermer [She91a] gave an $O(n)$ algorithm that solves the two-cover problem for convex polygons, and Belleville [Bel91] gave an $O(n^2)$ algorithm for the two-cover problem for L_k-convex polygons. We shall consider the two-cover problem for star-shaped polygons, and give an $O(n^4)$ time and $O(n^2 \log n)$ space algorithm to solve it (see also [Bel91]).

2 Geometric Considerations

2.1 Characterizing Kernel Points

Consider a polygon P that can be covered by two star-shaped polygons P_1 and P_2. In this section, we want to show that there exist star-shaped supersets P'_1, P'_2 of P_1, P_2 respectively, such that each of $Kr(P'_1)$, $Kr(P'_2)$ has at least one point whose location can be characterized. We first summarize the conditions under which a point of p is completely visible from $N_\varepsilon(v)$ for a vertex v of P and some $\varepsilon > 0$.

Lemma 2.1.1 *Let P be a simple polygon, v be a vertex of P, and p be a point of $V(P, v)$. There exists an $\varepsilon > 0$ such that p is completely visible from $P \cap N_\varepsilon(v)$ if and only if*

(a) p lies in the interior halfplanes determined by the edges of P containing v;

(b) No point of $bd(V(P, v))$ lies on the line segment (p, v) unless an edge of P containing v does.

We now want to obtain a characterization of the kernels of two star-shaped polygons that cover a simple polygon. First, we need to give a maximality criterion on which to base our choice of what constitutes a 'nice' cover of P by two star-shaped polygons.

Definition 2.1.1 *Let P be a simple polygon. We say that a star-shaped subset Q of P is maximal with respect to P if $Q = V(P, Kr(Q))$.*

From this definition of maximality, it follows immediately that, if P can be covered by two star-shaped polygons P_1 and P_2, then there are star-shaped polygons P'_1 and P'_2 that cover P and are maximal with respect to it. Contrarily to what happens when P is star-shaped, however. not every edge in $Kr(P'_1)$ and $Kr(P'_2)$ needs to be collinear with an edge of P. In spite of this, we can still locate at least one point in each of $Kr(P_1)$, $Kr(P_2)$ with respect to edges of P, as shown below.

A chord c of a simple polygon P shall be called the *extension* of a line segment l with respect to P if c contains l and no chord c' of P strictly contains c. In the remainder of this paper, we will simply refer to an *extension of P* to mean the extension of an edge of P. If p is a point of P, an edge e of P will be called *u-extreme* for p if the extension of e contains p, and if the point $p+u$ belongs to the exterior halfplane determined by e. A line segment l contained in P will be called an *internal tangent* to P at a vertex v_i of P if v_i belongs to the interior of l. We shall then say that the side of l to which the edges adjacent to v_i belong is an *exterior* halfplane determined by l.

Consider a simple polygon P, and a cover of P by an arbitrary number of star-shaped pieces $P_1,P_2,\dots,P_n$, each of which is maximal with respect to P. We can establish a relationship between the boundary of P and points of the boundary of $Kr(P_i)$.

Lemma 2.1.2 *Let P be a simple polygon covered by star-shaped polygons $P_1,P_2,\dots,P_n$ that are maximal with respect to P, u be a unit vector, and p be an extreme point of $Kr(P_i)$ in direction u. Either there is an edge e of P that is u-extreme for p, or there exists a reflex vertex v and a chord c that is an internal tangent to P at v, and is u-extreme for p.*

From Lemma 2.1.2 and from the previous observations, we get the following theorem that determines the possible locations of the kernel points of two star-shaped polygons covering P.

Theorem 2.1.1 *Let P be a simple polygon that can be covered by two star-shaped polygons. There exist two star-shaped polygons P_1, P_2 that cover P, and points $p_1 \in Kr(P_1)$, $p_2 \in Kr(P_2)$ that lie on extensions l_1, l_2 of P.*

A better characterization of kernel points would be desirable, as it would probably improve the algorithm which we will present. The following theorem, whose proof shall be omitted, shows that if such a characterization exists, it is unlikely to be a simple one :

Theorem 2.1.2 *There exists a simple polygon P such that, for every two-cover $\mathcal{P} = \{P_1,P_2\}$ of P by star-shaped polygons, no point of $Kr(P_1)\cup Kr(P_2)$ lies on the intersection of two lines determined by four not necessarily distinct vertices of P.*

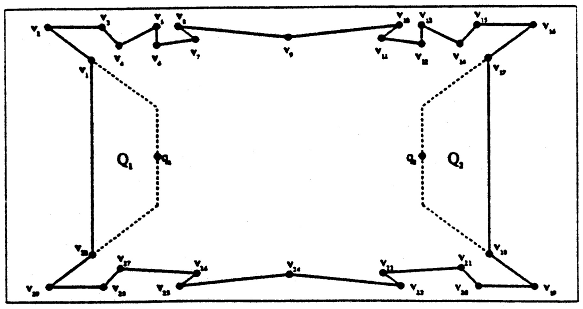

Figure 1: Polygon used in the proof of theorem 2.1.2

2.2 Linear-Rational Functions

This subsection examines the relation between visibility inside simple polygons and a class of functions that we shall call *linear-rational.*

Definition 2.2.1 *A function $f(t)$ is* linear-rational *if there exist real numbers a_1, a_2, a_3, a_4 such that $f(t) = (a_1t + a_2)/(a_3t + a_4)$.*

Consider a non-vertical directed line l in the plane, along with a reference point p_0 on l. Points of l will be parametrized by their signed distance from p_0. It is a well-known fact that, for each point p of a simple polygon P, and for every line segment l contained in P, the subset of l visible from p is connected. If l is weakly visible from p, but its left endpoint does not see p, then the line segment joining p to the leftmost point of l that is sees (denoted by *left*$[l,p]$) goes through a reflex vertex of P. We shall say that this vertex of P is *left-limiting* for p on l.

Suppose now that p moves along a line segment l' contained in P. The following lemma shows how to compute *left*$[l,p]$ for a given p provided that we know the vertex of P that is left-limiting for p on l.

Lemma 2.2.1 *Let l, l' be non-vertical lines in the plane, v be a point that does not belong to $l\cup l'$, and f be a linear-rational function of a parameter t. If t' is the position on l' of the point collinear with v and the point in position $f(t)$ on l, then t' is a linear-rational function of t.*

Thus, if l, l' are two line segments contained in P, for each vertex w_i of $P \setminus V(P,p)$ that is not a vertex of P, the position of w_i inside the edge of P to which it belongs is described by a piecewise linear-rational function of the position of p on l. Furthermore, the same lemma implies that the leftmost and rightmost points of l' visible from w_i are also described by this type of functions.

The positions of the points of l' that lie to the right of each one of *left*$[l',w_i]$ for a given position

t of p on l are described by the portion of the line $x = t$ that lies above the upper envelope of the set $F = \{f_1, f_2, \ldots, f_m\}$ of functions describing *left*$[l', w_i]$. Therefore, the set of points (t, t') above this upper envelope (where t is in the range over which each f_i describes *left*$[l', w_i]$) determines all pairs (p, p') of points for which p' lies to the right of *left*$[l', w_i]$ for each i. A symmetric fact holds for the lower envelope; these facts will be used in Section 3.3.

2.3 Visibility Changes Along a Line Segment

Consider a point p that belongs to a line segment l contained in a simple polygon P. As p moves along l, the set of vertices of $P \setminus V(P,p)$ remains the same at almost all points of l. In the neighborhood of these points, the intersection points of edges of P with edges of $V(P,p)$ that are chords of P move slightly in the interior of these edges of P. Since their movements are described by linear-rational functions of the position of p on l, the descriptions of these intersection points do not change and are easy to maintain. The changes in the set of vertices of $P \setminus V(P,p)$ thus occur at certain special points of l, which we shall call *critical*. These points are the following.

Definition 2.3.1 *Let P be a simple polygon, l be a line segment contained in P and p be a point of l. We say that p is a* critical point *of l if there is a ray R with endpoint p and a sequence $W = w_1, w_2, \ldots, w_k$ of two or more vertices of P such that the following conditions hold :*

- $R \cap l = p$;
- *For* $i = 1, \ldots, k$, $w_i \in R \cap V(P,p)$;
- *For* $i = 1, \ldots, k-1$, w_i *belongs to* $(p\ w_{i+1})$.

Each such w_i will be said to *generate* p. All changes in the equation describing the position of *left*$[e_k, p]$ (for some edge e_k of P) also occur at critical points of l. The next lemma shows that there are not too many critical points on each line segment.

Lemma 2.3.1 *Let P be a simple polygon and l be a line segment contained in P. There are at most $O(n)$ pairs (p, w), where p is a critical point of l, and w is a vertex of P generating p.*

Let us consider the way to cover $P \setminus V(P,p)$ as p moves along l. We need to determine whether a point of P sees all points of $P \setminus V(P,p)$. This can be done by noting that, if $\mathcal{Q} = \{Q_1, Q_2, \ldots, Q_k\}$ is a collection of simple subpolygons of P, then a point $p \in P$ sees all points of $\cup_{i=1}^{k} Q_i$ if and only if it sees all vertices of each Q_i.

We shall now restrict ourselves to points on extensions of P. We shall say that a line segment contained in a simple polygon P is *conforming* if both of its endpoints are vertices of P. Consider a conforming line segment l' contained in P. If a point p moves along a line segment l, all changes in the visibility of l' from p will occur at critical points of l (i.e. conforming line segments behave in the same way as edges of P). In general, extensions of P will not be conforming line segments. We can however add at most $2n - 6$ new vertices to P at the endpoints of these extensions; all extensions will then become conforming line segments. Using conforming line segments will simplify the algorithm by insuring that all important events occur at critical points of the augmented polygon.

3 Covering by Two Star-Shaped Polygons

In this section, we give an $O(n^4)$ time and $O(n^2 \log n)$ space algorithm to determine whether a given simple polygon can be covered by two star-shaped polygons, and to find two such polygons if they exist.

3.1 Algorithmic Preliminaries

We start by listing several computations which are used as preprocessing steps by the covering algorithm. Extensions of edges of P play an important role in our algorithm; we need a way to compute a list of all distinct extensions of edges of P. This can be done by $2n$ applications of the bullet shooting algorithm described in [GHL+87], using $O(n \log n)$ time and $O(n)$ space. Note that, contrarily to [GHL+87], we assume that bullet shooting returns the *last* point of $bd(P)$ that the ray meets before intersecting $ext(P)$, instead of the *first* point of $bd(P)$ that it meets. This simple modification affects neither the construction time nor the query time.

Another tool that we need is a way to determine, given a point p of P and an extension c_i of P, the subset of c_i that is visible from p, along with its limiting vertices. This can be done by modifying the algorithm used for problem (iv) in [GHL+87], which preprocesses P so that the subset of an edge e of P visible from a query point p can be found in $O(\log n)$ time. We divide P in two or more subpolygons by *cutting* it along c_i, and then preprocess each subpolygon individually.

A similar technique allows us to find the subset l of a line segment c_i that contains a given point p and such that, for all q in l, a given vertex v of P is a vertex of $V(P,p)$. This follows from the fact that v is a vertex of $V(P,p)$ if and only if p does not belong to the subset of P that completely sees an ε-neighborhood of v, and from Lemma 2.1.1.

Finally, one last application of this result allows us to determine in $O(\log n)$ time the subset of an extension c_i of P for which a vertex w of P is left-limiting (right-limiting) with respect to an edge e of P. This is because the range of allowable orientations for l can be found by intersecting the double wedges determined by the subsets of c_i and e that are visible from w. The preprocessing needed for the last three operation uses $O(n^2)$ time, with an $O(\log n)$ query time.

3.2 Two useful data structures

Consider a set $\mathcal{I} = \{I_1, I_2, \ldots, I_n\}$ of intervals, and the upper envelope over $I = \cap_{i=1}^{n} I_i$ of a set $\mathcal{F} = \{f_1(t), f_2(t), \ldots, f_n(t)\}$ of n linear-rational functions such that $f_i(t)$ is continuous over $I_i(t)$ for $i = 1, \ldots, n$.

It follows from a result of Hershberger [Her89] that the complexity of the upper envelope of $\mathcal{F}$ over I is $O(n)$. The algorithm described in [Her89] to merge two upper envelopes can be modified to merge the two upper envelopes corresponding to two sets $\mathcal{F}_1$ and $\mathcal{F}_2$ of linear-rational functions, in $O(|\mathcal{F}_1|+|\mathcal{F}_2|)$ time and space. By storing the functions of $\mathcal{F}$ in the leaves of a complete binary tree, we obtain the following theorem.

Theorem 3.2.1 *There exists a data structure which maintains the upper envelope of a set of linear-rational functions and such that each insertion and deletion takes $O(n)$ time, where n is the number of instances of functions stored in the data structure.*

We now define a geometric object that is used to find the critical points that belong to extensions of a simple polygon P.

Definition 3.2.1 *The* augmented arrangement *generated by a simple polygon P is the subdivision $\mathcal{S}$ of P generated by the extensions of its edges, where each vertex of P contains a pointer to the corresponding vertex of $\mathcal{S}$, and each edge of $\mathcal{S}$ holds a pointer to the corresponding extension of P.*

Since the step consisting in computing the augmented arrangement generated by P is not critical for our algorithm, it can be done using the algorithm of Bentley and Ottmann, and so the augmented arrangement generated by P can be computed in $O(n^2 \log n)$ time and $O(n^2)$ space.

To find the set P_i of critical points for each extension of a simple polygon P, and enumerate the points generating each of them, we look at each pair of vertices of P that may potentially generate critical points, and traverse the augmented arrangement $\mathcal{A}$ generated by P along the ray that they determine. Critical points of extensions of edges of P are located at the points of intersection of this ray with edges and vertices of $\mathcal{A}$. This procedure enumerates all critical points of extensions of p, and all points generating them, in $O(n^2 \log n)$ time using $O(n^2)$ space.

3.3 The Covering Algorithm

We now come to the description of the algorithm that finds a covering of P by two star-shaped polygons. We first explain the general idea behind the algorithm. We then detail each step of the algorithm, and conclude by sketching the proof of its correctness and deriving its running time.

3.3.1 General Idea

The algorithm is based on Theorem 2.1.1, which states that if P can be covered by two star-shaped polygons, then there are polygons P_1 and P_2 that cover P, and points $p_1 \in Kr(P_1)$, $p_2 \in Kr(P_2)$ that lie on extensions of P. We search each extension of P in turn for one such point, until we find it, or until all extensions of P have been visited.

The visit of an extension c_i proceeds by dividing it into intervals, whose endpoints are critical points of c_i; these endpoints are called *major events*. For every point p in the interior of a given interval, the equations describing the vertices of $P \setminus V(P,p)$ remain constant. For each vertex v of $P \setminus V(P,p)$, and for each extension c_j of P, v is visible from a connected subset c' of c_j. The equations describing the endpoints of c' do not need to remain constant between two major events on c_i; however they only changes when v crosses a critical point of the edge of P to which it belongs. The positions of p on c_i corresponding to these critical points are called *minor events*. Both kinds of events are stored in a heap that is continuously updated during the walk along c_i.

Major events are furthermore divided into two subtypes, called *left events* and *right events* respectively. Whenever a major event occurs at a point p of c_i, both a left event and a right event are added to the heap, except for the left and right endpoints of c_i, that only have right and left events respectively. A left event that occurs at a point p precedes the right event that occurs at this same point p. The point p is visited once the left event has been processed, but before the right event is taken care of. Intuitively, left events update the data structures from the situation that exists to the left of p to the situation at p, and right events update them from the situation at p to the situation to the right of p.

The algorithm visits all subintervals of c_i, and all of its critical points, from left to right. For each one, for each vertex v of $V(P,p)$, and for each extension c_j of P $(j \neq i)$, it computes the endpoints of the subset of c_j which sees v. A two-cover of P by star-shaped polygons is found when there are extensions c_i and c_j, and a point p of c_i such that the intersection, over all vertices v of $V(P,p)$, of the subset of c_j which sees v, is not empty.

Straightforward implementations of the above outline would lead to algorithms that run in $O(n^6)$ or $O(n^5 \log n)$ time. By updating incrementally the set of vertices of $P \setminus V(P,p)$, and the equations of the endpoints of the subsets of each c_j that see each of them, we can reduce this time to $O(n^4)$. The following subsections describe how this can be done.

3.3.2 Maintenance of $P \setminus V(P,p)$

We now describe the subroutines that maintain the portions of the data structures that are related to the vertices of $P \setminus V(P,p)$. These vertices can be subdivided into two classes : those that are vertices of P, and those that belong to the interior of edges of P.

For each extension c_j of P $(j \neq i)$, we maintain two trees of linear-rational functions : one contains the upper envelope of the functions describing *left*$[c_j, v]$ for each vertex v of $P \setminus V(P,p)$, and the other contains the lower envelope of the functions describing *right*$[c_j, v]$.

To add a vertex v of P to $P \setminus V(P,p)$, we first find the subset I of c_i which contains p and for which v is a vertex of $P \setminus V(P,p)$. We then add the endpoints of the subset of each c_j that sees v to the appropriate trees of linear-rational functions. Deleting v from $P \setminus V(P,p)$ only requires these nodes to be deleted from the trees.

Insertions of vertices of $P \setminus V(P,p)$ that are not vertices of P require more work, because the vertices of P that are left-limiting and right-limiting for them on extensions of P may change even when p is not a critical point of c_i. Thus the insertion subroutines need to add *minor events* to a heap (minor events are discussed in Section 3.3.3). Each edge e_k of P may contain up to two vertices of $P \setminus V(P,p)$ (*left*$[e_k, p]$ and *right*$[e_k, p]$). We only describe the insertion and deletion of *left*$[e_k, p]$, since *right*$[e_k, p]$ can be handled symmetrically.

To insert a vertex $v =$ *left*$[e_k, p]$ of $P \setminus V(P,p)$ into the data structures, we first compute the subset I of c_i that contains p and for which the position of v on e_k remains described by the same linear-rational function as at p. We then compute, for each extension c_j, the functions describing the left and right endpoint of the subset of c_j that remains visible from v as p moves inside I. These functions need not remain constant over the whole interval I, so we insert their values at p in the trees, and add minor events to the heap at the points (of c_i) at which they may change.

Deletions of vertices of $P \setminus V(P,p)$ are performed by removing from the trees corresponding to each extension c_j the nodes describing the endpoints of the subset of c_j from which the vertex of $P \setminus V(P,p)$ is visible.

Each of the procedures described in this subsection run in $O(n^2)$ time : each performs up to $O(n)$ insertions and deletions in a tree of linear-rational function, and each such operation takes $O(n)$ time by Theorem 3.2.1.

3.3.3 Major and Minor Events

A major event occurs at each critical point p for the extension c_i on which the point p is traveling. At each such point, one or several vertices may need to be inserted in $P \setminus V(P,p)$ or deleted from it. The intersection trees for all extensions of edges of P need to be updated accordingly. Throughout this subsection, we shall use the line of support of c_i as our axis of reference, with p moving from left to right on this line.

We only describe the manner in which left events are handled, since right events are taken care of symmetrically. This is done by visiting every vertex of P that generates p, and updating some pointers to vertices that are needed when this vertex is dealt with. We shall omit the detailed description of this operation. Every time that a vertex of $P \setminus V(P,p)$ appears or disappears, one of the procedures described in the previous subsection is used. Each such insertion or deletion can be done in $O(n^2)$ time.

Minor events occur when a vertex v of $P \setminus V(P,p)$, moving along an edge e of P as p moves along extension c_i, comes to a point that is critical for e. These are the points of e at which the equations describing the left and right endpoints of the subsets of extensions of P visible from v change. We need to visit each chord c_j for which the functions describing the endpoints of the subset of c_j that sees v change, and to update the trees containing these functions. Each such operation can be done in $O(n)$ time, since it involves a constant number of insertions and/or deletions in trees of linear-rational functions.

Once an event at a point p of c_i has been dealt with, we check each pair of trees that was updated for a solution. This can be done by finding a point that lies above the upper envelope corresponding to $left[c_j, w_i]$, and below the lower envelope corresponding to $right[c_j, w_i]$. If this point exists, then we obtain a pair of points of c_i, c_j whose visibility polygons cover P. Such a check takes $O(n)$ time for each extension that needs to be verified.

3.4 Correctness and Running Time

Using the lemmas and theorems that we developed in Section 2, we can show that, at each point as we move p on each c_i, the pair of trees corresponding to each c_j contain the functions describing the endpoints of the subsets of c_j that see each vertex of $P \setminus V(P,p)$.

Suppose that P can be covered using two star-shaped polygons. By Theorem 2.1.1, each of these polygons contains a kernel point that belongs to an extension of P. We eventually examine an interval containing such a point p; assume that it belongs to $Kr(P_1)$. We know that the point p' that is contained in $Kr(P_2)$ and belongs to an extension c_j sees all vertices of $P \setminus V(P,p)$. Thus the pair (p,p') lies above the upper envelope of the left endpoints of the subsets of c_j visible from each vertex of $P \setminus V(P,p)$, but below the lower envelope of the right endpoints of these subsets, and will be found. It is also clear that no anwer will be found unless one exists.

All preprocessing needed by our algorithm can be done in $O(n^2 \log n)$ time. Since only $O(n^3)$ events occur during the execution of our algorithm, the total time needed for heap operations in $O(n^3 \log n)$. Each extension c_i has only $O(n)$ critical points, at which $O(n^2)$ work needs to be done. Since $O(n^2)$ minor events will be handled during the traversal, and since each of them requires $O(n)$ time, the total amount of work that needs to be done for each extension that is visited is $O(n^3)$. The total size of the trees is $O(n^2 \log n)$. We have thus proved that

Theorem 3.4.1 *The outlined algorithm returns two star-shaped polygons P_1, P_2 covering the input polygon P if and only if two such polygons exist, and runs in $O(n^4)$ time using $O(n^2 \log n)$ space.*

4 Conclusion

We have given an algorithm to find a two-cover of a simple polygons by star-shaped polygons, if one exists, in $O(n^4)$ time and $O(^2 \log n)$ space. This result is probably not optimal, and finding how to improve it remains open.

The main difficulty that needed to be overcome to find this two-cover was that the kernel points do not need to be intersection points between two extensions of P. It is not clear if this is also the case when P is L_4-convex, but not L_3-convex. If not, the latter problem might be simpler.

It also remains to determine whether the methods that were used in this paper can be extended to solve covering problems for other small values of k, for instance for $k = 3$, or for polygons of link-radius j (a polygon P has link radius j if there is a point x of P such that, for each $p \in P$, there is a path between x and p using at most j edges).

Acknowledgements

The author thanks Godfried Toussaint for supervising this research, Tom Shermer for suggesting the problem that lead to it, and NSERC for its funding.

References

[Bel91] Patrice Belleville. Computing two-covers of simple polygons. Master's thesis, McGill University, Montréal, September 1991.

[GHL+87] L. Guibas, J. Hershberger, D. Leven, M. Sharir, and R. Tarjan. Linear Time Algorithms for Visibility and Shortest Path Problems Inside Triangulated Simple Polygons. *Algorithmica*, 2:209–233, 1987.

[Her89] J. Hershberger. Finding the Upper Envelope of n Line Segments in $O(n \log n)$ Time. *Information Processing Letters*, 33(4):169–174, 1989.

[O'R87] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Inc., 1987.

[She91a] T. C. Shermer. On Recognizing Unions of Two Convex Polygons and Related Problems. *Pattern Recognition Letters*, to appear.

[She91b] T. C. Shermer. Recent Results in Art Galleries. *IEEE Proceedings*, to appear.