A Simpler Circular Ray Shooting Algorithm

Ralph P. Boland[*†] Jorge Urrutia[‡]

Abstract

The circular ray shooting problem is to preprocess a simple polygon P so that, for a query directed circular arc ζ, one can quickly compute the first intersection point (if any) of ζ on P. We present a solution to this problem with the same preprocessing time, space, and query time complexities as the best known algorithm but our algorithm is simpler. Our algorithm is also more modular and uses tools for which we expect to find further applications. Our algorithm has the additional advantage that on a large class of polygons the expected space complexity is linear whereas the best known algorithm has an expected space complexity of $O(n \log n)$. We also provide a modification that allows the worst case preprocessing time of both our circular ray shooting algorithm and the best known algorithm to be reduced by an $O(\log n)$ factor.

1 Introduction

Let P be an n vertex simple polygon. Given a directed circular arc ζ, we denote the starting point of ζ by ζ_o and the endpoint of ζ by ζ_e. We use $\hbar_\zeta(P)$ to indicate the first point on $\zeta - \zeta_o$ to intersect the boundary of P in a directed traversal of ζ; if there is no such point then we give $\hbar_\zeta(P)$ the value ζ_e. We denote by $\zeta(P)$ the portion of ζ between ζ_o and $\hbar_\zeta(P)$, by $\mathrm{circle}(\zeta)$ the circle of which ζ is an arc, and by ζ_c the center of $\mathrm{circle}(\zeta)$. Consider the following problem:

Problem 1.1. Preprocess P so that, for a query arc ζ, with ζ_o interior to P, one can quickly determine $\hbar_\zeta(P)$.

We call this problem the circular ray shooting problem. For reasons that are explained in [5] and [3], we shall assume (without loss of generality) that the arcs used in this paper are x-monotone and directed from left to right with ζ_c below ζ, and that the angle of any tangent to the arc is between -$\pi/4$ and $\pi/4$.

The first solution to Problem 1.1 was given in [1] where it was shown that in $O(n \log^3 n)$ time a data structure requiring $O(n \log^3 n)$ space can be constructed with which a circular ray shooting query (henceforth called a query) can be performed on P in $O(\log^4 n)$ time. This result was improved upon in [5] with an algorithm in which P is preprocessed in $O(n \log^2 n)$ time to construct a data structure requiring $O(n \log n)$ space so that a query can be performed in $O(\log^2 n)$ time; this paper also provides a number of applications of circular ray shooting. We reported in [4] that [5] contains an error but this claim is based upon a misinterpretation and this error does not exist.

A major part of the results in [5] is a data structure, called a Hierarchical Vertical Decomposition, which is used to solve Problem 1.1. The main contribution of this paper is an alternate data structure that can be also be used to solve Problem 1.1 in the same complexities as in [5]. The main advantage of our data structure it that it is much simpler. It also has the advantage of being built from simpler components that we believe will find other applications. In addition, for a large class of polygons, the expected space complexity of our algorithm is reduced from $O(n \log n)$ to $O(n)$ while for this same class of polygons the expected space complexity of the algorithm in [5] remains at $O(\log n)$. We also provide a modification to one of the sub-algorithms in [5] that allows the preprocessing time of both the circular ray shooting algorithm in [5] and our algoirthm to be reduced by an $O(\log n)$ factor. Although we review some of the material in [1] and [5] we assume the reader is familiar with these papers.

In solving Problem 1.1 we use a much simpler version of the Hierarchical Vertical Decomposition data structure that is designed for monotone polygons. To apply our algorithm to P we use 2 new polygon partitioning algorithms that together partition P into x-monotone subpolygons such that any x-monotone curve interior to P, intersects at most $2\lfloor \log n \rfloor - 1$ subpolygons. This results in a circular ray shooting algorithm that is slower than the complexity we require by an $O(\log n)$ factor.

We remove this $O(\log n)$ factor using a new structure for applying range queries to trees; here we exploit that our partitioning algorithm produces a partition whose dual is a tree. The key reason that our algorithm is simpler despite needing these tools is that these tools are especially simple; alas the proofs that these tools work is more difficult and beyond the scope of this paper. The claims we make for our data structures and partitioning algorithms are established in [3] along with some applications.

For the remainder of this paper we will need some terminology. Let P be a simple polygon. We denote the first vertex of P by start(P); in this paper start(P)

*University of New Brunswick, Canada
email: rboland@unb.ca
†Funded in part by NSERC and the University of New Brunswick
‡Instituto de Mathematicas Universidad Nacional Autonoma de Mexico
email: urrutia@math.unam.mx

will always be a leftmost or rightmost vertex of P. We denote by $\Pi_\Box(P)$ the trapezoids of a vertical trapezoidization of P and by $\Gamma_\Box(P)$ the set of chords of P that induces $\Pi_\Box(P)$. For any element γ of $\Gamma_\Box(P)$, we denote by P_γ the subpolygon of P, of the 2 determined by γ, that does not contain start(P). Let Q be any subpolygon of P determined by a connected subset of $\Pi_\Box(P)$. We denote by door(Q) the chord of P on the boundary of Q such that Q is a subpolygon of $P_{\text{door}(Q)}$; if start(P) is a vertex of Q then Q has no door. We denote by windows(Q) the chords of P on the boundary of Q other than door(Q). Let R and S be 2 subpolygons of the partition of P by any subset of the elements of $\Gamma_\Box(P)$ such that door(S) $\in$ windows(R). Then we say that R is the *parent polygon* of S and S is a *child polygon* of R.

The remainder of this paper is organized as follows. In Section 2 we discuss a method of solving Problem 1.1 on an x-monotone polygon and its extension to general polygons. In Section 3 we discuss a method of partitioning P into x-monotone subpolygons such that any x-monotone curve interior to the polygon, intersects at most $2\lfloor\log n\rfloor - 1$ of the subpolygons. In Section 4 we discuss a method of solving Problem 1.1 where the query arc intersects as many as $2\lfloor\log n\rfloor - 1$ x-monotone subpolygons yet the cost is only a small constant factor higher than for an x-monotone polygon of the same size. This method uses the centroid path partition of a tree [6] and a new weighted binary tree structure [3] which is applied to each centroid path of the partition. In Section 5 we show how to reduce the preprocessing complexity of both the circular ray shooting algorithm in [5] and our algorithm to $O(n\log n)$. In Section 6, we discuss the construction of trapezoid sequences; Trapezoid sequences are useful for applications of circular ray shooting. Finally, in Section 7, we compare the size of the constants of the preprocessing, space, and query complexity of our algorithm and the algorithm in [5]. We establish that the size of these constants, in the worst case scenario for each algorithm, are roughly the same size . We also discuss a partial implementation of our algorithm. With this implementation we establish that, for a large class of polygons, the expected space complexity is linear for our algorithm, whereas the best case space complexity for this class of polygons for the algorithm in [5] remains at $O(n\log n)$.

2 Circular Ray Shooting in an X-Monotone Polygon

The data structures and algorithms in [5] and here for solving Problem 1.1 on an x-monotone polygon are nearly the same. The main differences between the method in [5] and our method are in how the x-monotone polygon case is generalized to handle simple polygons. In [5] the data structure itself is modified to deal with general polygons. We, instead, partition the polygon into x-monotone subpolygons and apply our data structure to each subpolygon.

Let Q be an h vertex x-monotone polygon whose leftmost points are on a vertical edge. We denote this edge by $\ell(Q)$. If the rightmost points of Q are on a vertical edge then we denote the interior of this edge by $r(Q)$; otherwise $r(Q) = \emptyset$. We call a circular arc ζ a *canonical* arc of Q if ζ_o belongs to the interior of $\ell(Q)$. For any canonical arc ζ of Q we call a test to determine if $\hbar_\zeta(Q) \in r(Q)$ a *canonical arc intersection test* on Q. If $\hbar_\zeta(Q) \notin r(Q)$ then the result of the test is *positive;* otherwise it is *negative.* The following result gives us a canonical arc intersection testing tool.

Lemma 2.1. *[5] In $\Theta(h\log h)$ time we can construct a data structure of size $\Theta(h)$ that allows any canonical arc intersection test on Q to be done in $\Theta(\log h)$ time.*

With this tool, and using divide and conquer, we can answer a query for Problem 1.1 on an m vertex x-monotone polygon in $O(\log^2 m)$ time. Let U be an m vertex x-monotone polygon.

The preprocessing stage involves the construction of a balanced rooted binary tree whose nodes are regions of U where each region is a connected subset of $\Pi_\Box(U)$. The leaves of the tree, from left to right, are the elements of $\Pi_\Box(U)$ ordered from left to right. Each nonleaf region R of the tree is the union of its 2 interior disjoint child regions. The root node is the region consisting of U. If 2 regions, say R_i and R_j, with disjoint interiors share a portion of their boundaries then we say that R_i is a neighbor of R_j. Each region has a link to an ancestor node of at most one of its neighbor regions to its right.[1] We call these links *neighbor ancestor links.* The tree has $\lceil\log m\rceil$ levels and each the regions of each level contain all the trapezoids of $\Pi_\Box(U)$. Thus $\Theta(m\log m)$ space is required in total.

We process a query arc ζ of U where some trapezoid R_0 of $\Pi_\Box(R)$ contains ζ_o as follows. First, R_0 is found, if not already known. This can be done in $O(\log n)$ time after a linear time preprocessing stage using point location [1]. Next, $\hbar_\zeta(R_0)$ is determined and if $\hbar_\zeta(U) = \hbar_\zeta(R_0)$ we are done. Otherwise a link at R_0 to a neighbor ancestor region, say R_1, of R_0 is followed and the maximal sub-arc, say ζ_1, of ζ that is canonical with respect to R_1 is determined. No part of ζ_1 is to the left of $\ell(R_0)$. A canonical arc intersection test is now performed with ζ_1 on R_1. If this test is negative then this process is repeated with R_1 replacing R_0 and in general R_{i+1} replacing R_i until a region,

[1] In practice, each region would also have a link to an ancestor node of at most one of its neighbor regions to its left since this would allow circular ray shooting with counterclockwise oriented arcs and with arcs on the bottom quarter of a circle.

say R_k, of the tree is found such that the canonical arc intersection test on R_k is positive. Now $\hbar_\zeta\,(U)$ is found by recursively following the link to the child region of R_k containing $\hbar_\zeta\,(U)$, as determined by a canonical arc intersection test of the left child, until the trapezoid of $\Pi_\square(U)$ containing $\hbar_\zeta\,(U)$ is found. From this trapezoid $\hbar_\zeta\,(U)$ is determined. Since the tree has at most $\lceil \log m \rceil$ levels, at most $2\lceil \log m \rceil - 1$ canonical arc intersection tests are required to find $\hbar_\zeta\,(U)$ and thus the query time is $O(\log^2 m)$.

In [5] this tree structure is modified to work for general polygons. Among the changes needed, a region may be represented by as many as 2 x-monotone polygons. The percentage of the regions represented by 2 x-monotone polygons is very roughly the percentage of elements of $\Pi_\square(P)$ with more than 2 adjacent trapezoids.

Our algorithm, instead, partitions P into x-monotone subpolygons. There is then a tree for each x-monotone polygon in the partition.

3 X-Monotone Subpolygon Partitions

In this section we show how to partition P into x-monotone subpolygons such that any x-monotone curve interior to P intersects at most $2\lfloor \log n \rfloor - 1$ of the subpolygons. This is done in 2 stages. We call a polygon T *tree* $\pm x$*-monotone* if, for any point q interior to T, there is an x-monotone curve contained in T from start(T) to q; if start(T) is a leftmost vertex of T then we call T *tree* $+x$*-monotone* and if start(T) is a rightmost vertex of T then we call T *tree* $-x$*-monotone.* In the first stage we partition P into $\pm x$-monotone subpolygons using an algorithm in [3]; we denote this partition by $\Pi_{\mathcal{T}}(P)$. The algorithm runs in $O(n)$ time and is surprisingly simple, given $\Pi_\square(P)$. A longer but clearer variation of this algorithm is as follows.

Construct the dual tree of $\Pi_\square(P)$ such that the vertices are contained in the trapezoids they correspond to and the edges are x-monotone curves interior to P. Set the root node of the tree to the node whose dual trapezoid contains start(P) and direct the edges of the tree towards the root. Color each edge of the tree blue if it is directed to the left and red if it is directed to the right. Give each vertex of the tree the same color as its only outward directed incident edge except the root vertex which is given the color of its only incident edge. Let ξ be the set of edges of the tree whose endpoints are colored differently. Let $\Gamma_{\mathcal{T}}(P)$ be the subset of $\Gamma_\square(P)$ whose elements are crossed by some element of ξ. Note that $|\xi| = |\Gamma_{\mathcal{T}}(P)|$. Then $\Pi_{\mathcal{T}}(P)$ is the partition of P induced by $\Gamma_{\mathcal{T}}(P)$.

In [3] the following property of $\Pi_{\mathcal{T}}(P)$ is established;

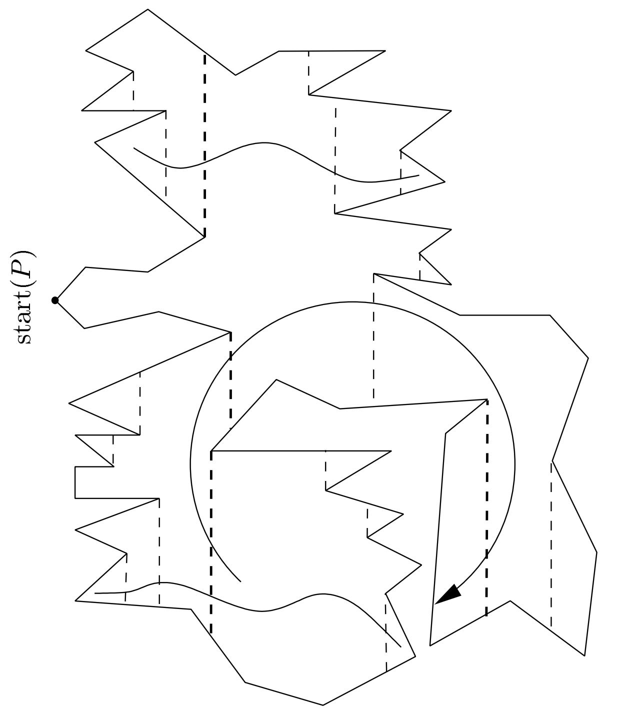

Figure 1: Any x-monotone curve interior to a polygon P intersects at most 2 elements of $\Pi_{\mathcal{T}}(P)$ (separated by thick dashed lines) and $2\lfloor \log n \rfloor - 1$ elements of $\Pi_{\nmid}(P)$ (separated by all dashed lines) where $|P| = n$. Also, any directed circular arc, interior to P but otherwise unconstrained, intersects at most 4 elements of $\Pi_{\mathcal{T}}(P)$.

see Fig. 1:

Lemma 3.1. *Any x-monotone curve interior to P intersects at most 2 elements of* $\Pi_{\mathcal{T}}(P)$.

Lemma 3.1 essentially reduces Problem 1.1 to the case that the polygon is tree $\pm x$-monotone; we can then apply the data structure we construct for tree $\pm x$-monotone polygons to each element of $\Pi_{\mathcal{T}}(P)$. For example, say, for a query arc ζ, that ζ_o is contained in a tree $\pm x$-monotone polygon, say Q, of $\Pi_{\mathcal{T}}(P)$. First, we find $\hbar_\zeta\,(Q)$. Now if $\hbar_\zeta\,(Q)$ belongs to the interior of an element γ of $\Gamma_{\mathcal{T}}(P)$ then we use the sub-arc of ζ that has $\hbar_\zeta\,(Q)$ and ζ_e as its endpoints for a new query on the element of $\Pi_{\mathcal{T}}(P)$, other than Q, for which γ is either a door or a window. By Lemma 3.1, the second query must find $\hbar_\zeta\,(P)$.

Next we apply the second stage of our polygon partitioning algorithm. Let T be an m vertex element of $\Pi_{\mathcal{T}}(P)$; we partition T into x-monotone subpolygons such that any x-monotone curve interior to T intersects at most $\lfloor \log m \rfloor$ of the subpolygons. We use a method analogous to the construction of the centroid path partition of a tree [6]; see also [3]. For each trapezoid t in $\Pi_\square(T)$ with a window we mark all the windows of t except the window, say w_t, of t for which $|\Pi_\square(T_{w_t})|$ is maximum with ties broken arbitrarily; we denote this set of marked chords of $\Pi_{\square}(T)$ by $\Gamma_{\lightning}(T)$. We further denote by $\Pi_{\lightning}(T)$ the partition of T induced by $\Gamma_{\lightning}(T)$. The following 2 properties of $\Pi_{\lightning}(T)$ are used in this paper and are easily established; see [3]:

Lemma 3.2. *Each element of* $\Pi_{\lightning}(T)$ *is* x*-monotone.*

Lemma 3.3. *Let* S *and* γ *be such that* $S \in \Pi_{\lightning}(T)$ *and* $\gamma \in$ *windows*(S)*. Then* $2|\Pi_{\square}(T_\gamma)| < |\Pi_{\square}(T_{\mathrm{door}(S)})|$.

Using Lemma 3.2, we can now solve Problem 1.1 on T. First, we apply the preprocessing stage of the algorithm from Section 2 to each x-monotone subpolygon in $\Pi_{\lightning}(T)$. Then, given a query arc ζ, we apply the query stage of the algorithm from Section 2 to each x-monotone subpolygon in $\Pi_{\lightning}(T)$ intersected by ζ in succession (as described above for arcs intersecting 2 elements of $\Pi_{\mathcal{T}}(P)$). Using Lemma 3.3, it can be shown that $\zeta(T)$ intersects at most $\lfloor \log m \rfloor$ elements of $\Pi_{\lightning}(T)$. Using Lemma 2.1 and divide and conquer, we can answer a query for Problem 1.1 on an h vertex x-monotone polygon in $O(\log^2 h)$ time. Thus we can answer a query on T in $O(\log^3 m)$ time. Therefore, by Lemma 3.1, we can answer a query on P in $O(\log^3 n)$ time. This is slower than the complexity we require by an $O(\log n)$ factor.

4 Circular Ray Shooting in a Tree $\pm X$-Monotone Polygon

We reduce the cost of a circular ray shooting query on T to $O(\log^2 m)$ time, and thus answer a query on P in $O(\log^2 n)$ time, by using an algorithm for performing a kind of range query on trees; see [3].

Our first step is to assign weights to the elements of $\Pi_{\square}(T)$. For each element t_i of $\Pi_{\square}(T)$ we assign a weight $w(t_i)$ as follows: if t_i has no door or if $\mathrm{door}(t_i) \in \Gamma_{\lightning}(T)$ then $w(t_i) = 1$; otherwise $w(t_i) = \sum_{j=1}^{k} |\Pi_{\square}(P_{\gamma_j})|$ where $\gamma_1, \gamma_2, ..., \gamma_k$ are the windows of the parent of t_i other than $\mathrm{door}(t_i)$.

The next step is to construct a version of the tree data structure described in Section 2 for each element of $\Pi_{\lightning}(T)$. Each tree is constructed based on the augmented weighted binary tree structure described in [3]. For any x-monotone subpolygon Q of $\Pi_{\lightning}(T)$ we use $\mathbb{T}(Q)$ to denote the data structure we construct for Q and $W(Q)$ to denote the sum of the weights of the elements of $\Pi_{\square}(Q)$. Each node of $\mathbb{T}(Q)$ is assigned a weight which is the sum of the weights of the leaves of the subtree rooted at the node. The tree component of the augmented weighted binary tree structure is constructed using 2 stacks and at most $8|\Pi_{\square}(Q)|$ push/pop operations. The augmented part corresponds to the set of neighbor ancestor links of $\mathbb{T}(Q)$ and is also easily implemented.

We now assume that T is tree $+x$-monotone; the alternate case is handled only slightly differently; see [3]. For a given query arc ζ of Q, let $w_{\hbar}$ be the weight of the trapezoid in $\Pi_{\square}(Q)$ immediately following the one containing $\hbar_\zeta(Q)$ or the rightmost trapezoid in $\Pi_{\square}(Q)$ if it contains $\hbar_\zeta(Q)$. Let w_ζ denote the total of the weights of the trapezoids of $\Pi_{\square}(Q)$ intersected by $\zeta(Q)$. In [3], it is shown that, using $\mathbb{T}(Q)$, we can find $\hbar_\zeta(Q)$ in $O(\max(1, \log w_\zeta - \log w_{\hbar}))$ canonical arc intersection tests. Let $\zeta' = \zeta - \zeta(Q)$. Now consider that $\hbar_\zeta(Q)$ belongs to the interior of a window of Q. Let U be the element of $\Pi_{\lightning}(T)$ whose door is this window and $w'_{\hbar}$ be the weight of the trapezoid in $\Pi_{\square}(U)$ immediately following the one containing $\hbar_{\zeta'}(U)$ or the rightmost trapezoid in $\Pi_{\square}(U)$ if it contains $\hbar_{\zeta'}(U)$. Then, in [3], it is shown that $\hbar_\zeta(Q)$ and then $\hbar_{\zeta'}(U)$ can be found in

$$\begin{aligned}&\Theta(\max(1, \log w_\zeta - \log w_{\hbar}) + (1, \log w_{\hbar} - \log w'_{\hbar}))\\&= \Theta(2, \log w_\zeta - \log w'_{\hbar})\end{aligned}$$

canonical arc intersection tests. Note that if the dual of $\Gamma_{\lightning}(T)$ is a binary tree then $\log w_{\hbar} = W(U)$. Similarly, no matter how many elements of $\Pi_{\lightning}(T)$ are intersected by ζ before intersecting T, the total number of canonical arc intersection tests performed is never more than $O(\log W(Q)) = O(\log n)$. A key point here is that Lemma 3.3 guarantees that ζ intersects at most $\lfloor \log m \rfloor$ elements of $\Pi_{\lightning}(T)$ so the overhead of switching between x-monotone subpolygons is not a cost factor.

5 Reducing the Preprocessing Time Complexity to $\Omega(n \log n)$

In this Section we show how to reduce the preprocessing complexity of both our solution to Problem 1.1 to $\Theta(n \log n)$. This algorithm is a modification of an algorithm in [5] and can also be used to reduce the preprocessing complexity of the solution in [5] to $\Theta(n \log n)$ as well. Let $M = u_0, \ldots, u_{m-1}$ be an m vertex x-monotone chain. Given a directed circular arc ζ, we use $\hbar_\zeta(M)$ to indicate the first point on ζ to intersect M in a directed traversal of ζ; if there is no such point then we give $\hbar_\zeta(M)$ the value ζ_e. We denote by $\zeta(M)$ the portion of ζ between ζ_o and $\hbar_\zeta(M)$. Consider the following problem:

Problem 5.1. Preprocess M so that, for any query arc ζ, with ζ_o and ζ_e interior to circle(ζ), with ζ_o on the vertical line through u_0 and above u_0, and with ζ_e on the vertical line through u_{m-1} and above u_{m-1}, one can quickly determine if ζ intersects $\hbar_\zeta(M)$.

This problem is solved in $O(m \log m)$ preprocessing time, $O(m)$ space, and $O(\log m)$ query time in [5]. In this section we show that the preprocessing time for this problem can be reduced to $\theta(m)$ without changing the space complexity or query complexity. Using this

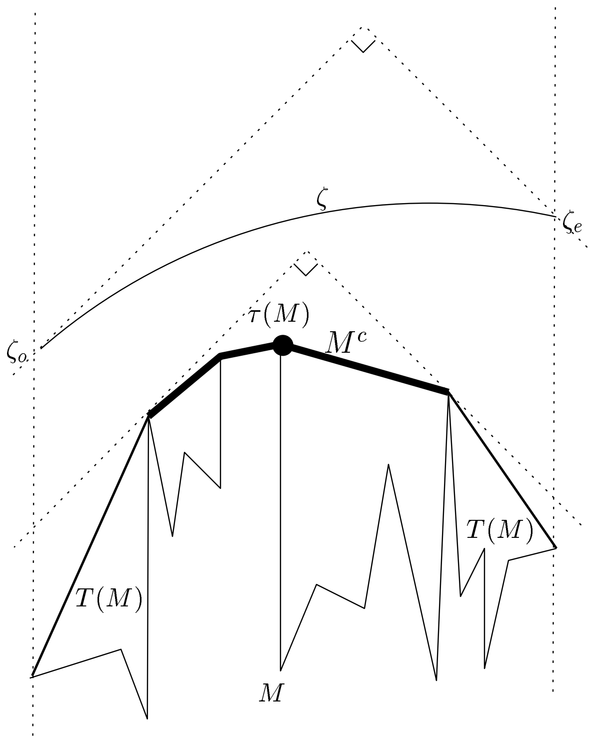

Figure 2: Subchain M^c of x-monotone chain M

improvement reduces the preprocessing complexity of our circular ray shooting algorithm to $O(n \log n)$. We point out if that ζ was permitted to be a full circle then we could solve this problem, in the stated complexities, by constructing a farthest point Voronoi diagram of the vertices of M. Then, using this diagram, we could find the farthest vertex, say v_ζ, of M from circle(ζ). Now circle(ζ) would intersect M if and only if v_ζ is on or interior to circle(ζ). But, since ζ is not a circle, we have to deal with the fact that some vertices of M may be exterior to circle (ζ) even though ζ does not intersect M; these vertices would be below circle (ζ). The solution to this problem, given in [5], is to eliminate from M any vertices that could intersect circle (ζ) but not ζ. This is possible to do, without knowing ζ, because of the constraint that ζ be a sub-arc of the top quarter of a circle .

The preprocessing complexity of Problem 5.1 can be reduced to $\theta(m)$ in a simple manner. First construct, in $\Theta(m)$ time the convex hull of M. Then apply the method used in [5] to eliminate from the convex hull any vertices that may be below circle (ζ) (without knowing ζ). Now construct the farthest point Voronoi diagram of the remaining vertices of the convex hull. This can be done in $\Theta(m)$ time because of two facts. First, the Voronoi diagram of the vertices of a convex polygon can be constructed in linear time [2] and second, the problem of constructing the farthest point Voronoi diagram of a point set can be transformed into the problem of constructing a Voronoi diagram of the same set in linear time [7].

The farthest point Voronoi diagram so constructed may still contain some unnecessary sites. We can eliminate all unnecessary vertices (for Problem 5.1) as follows.

Let H be the convex hull of M. Assume that H is directed counterclockwise and its edges are directed accordingly. We call the subchain of H whose edges have directions between $\pi/2$ and $3\pi/2$ the left hull of M and denote it by $T(M)$. We denote by M^c the subchain of H whose edges have directions strictly between $3\pi/4$ and $5\pi/4$. See Fig 2.

Let $\tau(M)$ denote the highest vertex on M with ties broken by (arbitrarily) choosing the leftmost vertex. Let $\mathbb{V}$ denote the farthest point Voronoi diagram of M^c. Let radius(ζ) denote the radius of circle (ζ) and $d(M, \zeta)$ be the distance from center(ζ) to the vertex of M that is farthest from center(ζ).

Now, if ζ is above the horizontal line through $\tau(M)$ then clearly ζ does not intersect M. If, instead, any part of ζ is below the vertical line through $\tau(M)$ then we claim that ζ intersects M if and only if:

$$d(M^c, \zeta) \geq \text{radius}(\zeta) .$$

If this claim were untrue it could fail for only two reasons:

1. Some vertex v on $T(M)$ is above ζ yet $d(M^c, \zeta) <$ radius(ζ). Then v is on a subchain, say C, of $T(M)$ such that exactly one vertex (an endpoint) of C, say p, is a vertex of M^c and the other endpoint, say q, of C is an endpoint of $T(M)$. For any point t on C let $d(t)$ be the vertical distance from t to a point on ζ where this distance is negative if t is above ζ. Then $d(p) > 0$ since $d(M^c, \zeta) <$ radius(ζ). Furthermore, by the construction of C and the constraints on ζ, the slopes of the tangents of ζ and C are such that the function d is monotonically increasing as we move along C from p to q. But this contradicts that $d(v) \leq 0$. See Fig. 3.

2. Some vertex u on $T(M^c)$ is below circle (ζ) yet $\zeta \cap M = \emptyset$. Assume, without loss of generality, that u is to the right of $\tau(M)$. Let λ_v be the vertical line through ζ_e. Let p_τ be the intersection of the horizontal line through $\tau(M)$ and λ_v. Let λ_τ be the line through $\tau(M)$ with a slope of -1 and let q_τ be the intersection point of λ_τ with λ_v. Let $\overline{\zeta}$ be the extension of ζ to a full top quarter circle. Let $p_{\overline{\zeta}}$ be the intersection point of the horizontal line through the endpoints of $\overline{\zeta}$ with λ_v. Let p_ζ be the point where circle (ζ) intersects λ_v for the first time and q_ζ be the point where circle (ζ) intersects λ_v for the second time where we traverse circle (ζ) in the clockwise direction starting from ζ_o. Let d_x

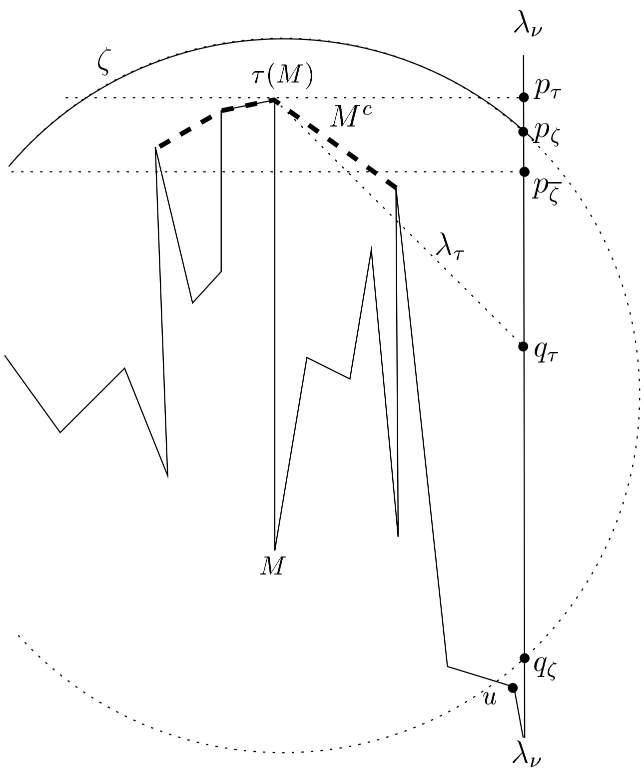

Figure 3: If part of ζ is below τ(M) then no vertex on M is above ζ if and only if no vertex on M^c is above circle (ζ).

be the distance between vertical lines through the endpoints of M. For any point p let $y(p)$ denote the y-coordinate of p. Now, by the constraints on ζ, the distance from $p_{\overline{\zeta}}$ to q_ζ is at least d_x. Also, by construction, the distance from to p_τ to q_τ is at most d_x. But then, since $y(p_\tau) \geq (p_\zeta) \geq p_{\overline{\zeta}}$, it must be that $y(q_\tau) \geq y(q_\zeta)$. But, since $u \neq v_{m-1}$, clearly u is above λ_τ so u cannot be below circle (ζ) since then it would be below the bottom quarter of circle(ζ) which is below q_ζ. See Fig. 3.

Note that if ζ were not constrained to be a subarc of the top quarter arc of a circle then this claim would not in general hold; thus the need for this constraint.

Our algorithm for determining which points of M to include in the farthest point Voronoi diagram is optimal in the following sense.

Lemma 5.2. *For every site s_i of $\mathbb{V}$ there exists an arc ζ_i, satisfying the constraints of Problem 5.1, such that s_i is the only vertex of M that is on or above ζ_i.*

Proof. Let v be any vertex of M^c and let l be the line through the midpoints of the 2 edges of M^c incident to v (The case that v has fewer than two incident vertices can be handled similarly.). Then l is an arc of infinite radius, every vertex of M^c is below l except v, and, by construction, l satisfies the constraints of Problem 5.1. $\square$

We can further reduce the cost of constructing all the farthest point Voronoi diagrams needed for solving the circular ray shooting problem from $O(n \log n)$ to $O(n + r \log r)$ where r is the number of sites used in all the farthest point Voronoi diagrams; see [5].

6 Trapezoid Sequences

In [5], the applications given for circular ray shooting use the following result.

Lemma 6.1. *[5] Let P be a simple n vertex polygon. There is a data structure of size $O(n \log n)$ that can be computed in $O(n \log n)$ time that returns in $O(\log n)$ time, the sequence of trapezoids connecting 2 query trapezoids t_1 and t_2 represented as a sequence of $O(\log n)$ precomputed sequences.*

We can extend our algorithm to also provide this capability but not without additional efforts that would undermine the advantages we have gained over the algorithm in [5]. However, for the applications of circular ray shooting provided in [5], the query points are required to be connect-able by a circular arc interior to P (for example finding the shortest circular arc interior to a polygon joining 2 query points). Thus, for these applications, we only require a version of Lemma 6.1 that works for tree $\pm x$-monotone polygons and this can be done by extending our data structures in the same manner as in [5]. This is sufficient because a circular arc interior to P can intersect at most 4 elements of $\Pi_{\mathcal{T}}(P)$ (see Fig. 1 and [3]) so at most 4 tree $\pm x$-monotone subpolygons of $\Pi_{\mathcal{T}}(P)$ would need to be processed to construct such a trapezoid sequence.

7 Complexity Constants

Since both the solution to Problem 1.1 in [5] and our solution have the same asymptotic complexities, a comparison of the constants for these complexities is worthwhile. Let P be an n vertex simple polygon and k be the size of the largest element in $\Pi_{\mathcal{T}}(P)$ by vertex count. For our algorithm the preprocessing time and space complexity is $\Theta(n \log k)$ and the query complexity is $O(\log^2 k)$. The same asymptotic complexities can be established for the circular ray shooting algorithm in [5]. Thus we restrict our analysis to the case that P is tree $\pm x$-monotone.

We can compare the query complexity constants by counting the number of canonical arc intersection tests performed which, in the worst case scenario, which, in

[5], is determined by the height of the tree constructed and which, for our algorithm, is determined by the height of the largest weighted binary tree constructed. In [5], the authors determined an upper bound on the height of the tree constructed by their algorithm of $2.41 \log n$.[2] We have determined an upper bound on the height of the largest weighted binary tree constructed by our algorithm of $2.52 \log n + 2$.[3] We gratefully thank J. D. Horton for improving this bound from the original $3 \log n + 2$.

We can compare the preprocessing time and space complexity constants by counting the total number of trapezoids in the vertical trapezoidizations of the monotone subpolygons stored at the nodes of the trees described in Section 2. Using the height of the tree (or weighed binary trees for our algorithm) we get an upper bound of $4.82n \log n$ on the number of trapezoids used for the algorithm in [5] (we think the correct value is around $(3/2)n \log n$) and $2.52n \log n + 2n$ for our algorithm. However, in both cases, these bounds are clearly not tight. The best case scenario for the algorithm in [5] is that P is x-monotone. Then, between $n \log n$ and $(n+1) \log(n+1)$ trapezoids are used. The same number of trapezoids are used in this case with our algorithm and we conjecture that it is the worst case scenario. The best case scenario for our algorithm is that the dual of $\Pi_{\square}(P)$ is a complete perfectly balanced tree. Then our algorithm uses fewer than $5n/2$ trapezoids.

In order to determine the complexity constants of our algorithm more accurately we have implemented the algorithm for performing range searches on trees. This includes constructing the centroid path decomposition of the tree, the construction of weighted binary trees on each path and the algorithms for performing queries on the tree. Instead of the circular ray shooting problem we solve a problem relating to intervals stored at the nodes of the tree; see the implementation or [3]. Instead of counting trapezoids we sum of the external path lengths[4] of the weighted binary trees constructed by our algorithm; it is easily shown that these numbers are the same.

The implementation consists of approximately 300 lines of C language code exclusive of comments, blank lines, and test code. The results of our tests are preliminary. We considered the case of random insertion binary trees and random binary trees. We believe that for both these cases the expected sum of the external path lengths is $O(n)$ and our experiments support this claim. In the random insertion binary tree case our experiments indicate that the expected sum of the external path lengths is close to $3.2\,n$. In the random binary tree case our experiments indicate that the expected sum of the external path lengths is close to $4.6\,n$. We also tested a variation of random insertion binary tree in which at each step on an insertion we branch left or right with equal probability. For this case the expected sum of the external path lengths is close to $3.0\,n$. Further information on these experiments may be found on the web page of the first author.

[2] We do not believe that this value is tight. We think it can be reduced to approximately $1.71 \log n$.

[3] Since an arc may intersect as many as 2 elements of $\Pi_{\mathcal{T}}(P)$ we should multiply this value by 2. However, reorganizing the weights assigned to nodes of the tree can eliminate this factor. This change may not be worthwhile since it may decrease the average case performance.

[4] The external path length of a tree is the sum of the number of vertices of all the paths from each leaf of the tree to the root.

References

[1] Pankaj K. Agarwal and Micha Sharir. Circle shooting in a simple polygon. *J. Algorithms*, 14:69–87, 1993.

[2] A. Aggarwal, Leonidas J. Guibas, J. Saxe, and P. W. Shor. A linear-time algorithm for computing the Voronoi diagram of a convex polygon. *Discrete Comput. Geom.*, 4(6):591–604, 1989.

[3] R. P. Boland. *Polygon Visibility Decompositions with Applications*. Ph.D. thesis, submitted January, 2001, University of Ottawa, Ottawa, Ontario.

[4] R. P. Boland and J. Urrutia. Circular ray shooting: Slower but surer. In *Proc. 12th Canad. Conf. Comput. Geom.*, pages 37–42, 2000.

[5] Siu-Wing Cheng, Otfried Cheong, Hazel Everett, and René van Oostrum. Hierarchical vertical decompositions, ray shooting, and circular arc queries in simple polygons. In *Proc. 15th Annu. ACM Sympos. Comput. Geom.*, pages 227–236, June 1999.

[6] Richard Cole, Martin Farach-Colton, Ramesh Hariharan, Teresa Przytycka, and Mikkel Thorup. An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees. *SIAM J. Comput.*, 30(5):1385–1404 (electronic), 2000.

[7] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, October 1990.