Circular Ray Shooting: Slower but Surer

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. In a paper presented at the 15th A.C.M. Symposium on Computational Geometry Cheung et. al. showed that the circular ray shooting problem can be solved with $O(\log^2 n)$ query time after $O(n \log^2 n)$ preprocessing and using a data structure requiring $O(n \log n)$ space. Unfortunately there is an error in their algorithm. We report on this error and provide a correction. Our correction comes with a price; the query time complexity and space complexity are both increases by a $O(\log n)$ factor. In spite of these extra costs the modified algorithm is the fastest known for this problem.

An alternative to the correction we provide here is to use a different Voronoi diagram. This diagram reduces the cost of a query from $O(\log^3 n)$ back to $O(\log^2 n)$. We show however that, in the worst case, this change may increase the storage requirements to $O(n^2 \log n)$.

1 Introduction

Let P be a simple polygon with n vertices. 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 ζ 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 circle(ζ) the circle of which ζ is an arc, and by ζ_c the center of circle(ζ). Consider the following problem:

Problem 1.1. Preprocess P so that, for a query arc ζ, where ζ_o is in the interior of P, one can quickly determine $\hbar_\zeta(P)$.

We call this problem the circular ray shooting problem. The first solution for Problem 1.1 was given in [1] where it was shown that in $O(n \log^3 n)$ time we can construct a data structure requiring $O(n \log^3 n)$ space which can be used to find $\hbar_\zeta(P)$ in $O(\log^4 n)$ time. This result was recently improved upon in [2] with an algorithm in which a polygon is preprocessed in $O(n \log^2 n)$ time to construct a data structure requiring $O(n \log n)$ space so that a circular ray shooting query can be performed in $O(\log^2 n)$ time. However the paper contains an error. We present here a fix for this error. Unfortunately the correction adds an $O(\log n)$ factor to each of the query time complexity and space complexity of the algorithm in [2]. Nevertheless, with this correction applied, the algorithm in [2] remains the fastest known solution for Problem 1.1. We point out that this difficulty does not occur in [1].

We assume wlog that ζ is an x-monotone circular arc directed from left to right with ζ_c below ζ. In [2] this assumption is based on the fact we can cut ζ into at most five components, each of which is either the top quarter circle, the bottom quarter circle, the left quarter circle, or the right quarter circle of circle(ζ) or a portion thereof. We can then solve the circular ray shooting problem for each component of ζ in turn in the order implied by the direction of ζ. For each component, appropriate reflections of both the component and P about the x-axis, the y-axis, the line $y = x$ or combinations of these reflections are applied, if necessary, so that the component satisfies the stated assumption.

In this paper we instead base the assumption on the fact that we can cut ζ into at most three components, each of which is either the top semi-circle or bottom semi-circle of circle(ζ) or a portion thereof. For each component we apply reflections about the x-axis, the line $y = x$, or both as necessary so that the component satisfies the stated assumption[1]. We change the way this assumption is justified because our fix requires it, though it also makes matters simpler. We point out that ζ is more tightly constrained in [2] than the constraints we have applied. However the additional constraints do not prevent the error that we are dealing with here. The changes to the main data structure in [2] as a consequence of this change are straightforward and we omit the details.

Although we review some of the material in [1] and [2] we assume that the reader is familiar with these papers.

*S.I.T.E. University of Ottawa, Canada
email: rboland@site.uottawa.ca

†Instituto de Mathematicas Universitidad Nacional Autonoma de Mexico
email: urrutia@math.unam.mx

[1] Actually the data structure we use supports arcs directed in both clockwise and counterclockwise directions so that a reflection about the line $y = x$ is not necessary.

2 The Error

In [2] is defined a data structure on polygon P called the hierarchical vertical decomposition of P and which we denote by $\mathcal{H}$. In all, $\mathcal{H}$ requires $O(n \log^2 n)$ time to construct and it contains $O(\log n)$ partitions of P into subpolygons in a hierarchical structure. The total number of vertices of the subpolygons in each partition is $O(n)$ so, altogether, there are $O(n \log n)$ vertices in the subpolygons of P stored in $\mathcal{H}$. Other parts of $\mathcal{H}$ also require $O(n \log n)$ storage so the structure requires $O(n \log n)$ space in total.

The edges of the subpolygons in $\mathcal{H}$ that are not edges or parts of edges of P are called doors. Any subpolygon of $\mathcal{H}$ contains at most three doors all of which are vertical. A circular arc may cross from one subpolygon of $\mathcal{H}$ to another without intersecting P by crossing doors.

Using $\mathcal{H}$, the circular ray shooting algorithm in [2] reduces the circular ray shooting problem to $O(\log n)$ instances of Problem 2.1.

Let d_1 and d_2 be two doors of $\mathcal{H}$ that are edges of a subpolygon Q of $\mathcal{H}$. For reasons that are explained in [2] we may assume that Q is an x-monotone polygon. Assume, for some sub-arc $\overline{\zeta}$ of ζ, that $\overline{\zeta}_o \in d_1$ and $\overline{\zeta}$ is directed towards the interior of Q at $\overline{\zeta}_o$. Assume furthermore that $\overline{\zeta}_e \in d_2$ and $\overline{\zeta}$ is directed towards the exterior of Q at $\overline{\zeta}_e$. The case that $\overline{\zeta}$ intersects d_1 or d_2 tangentially must also be dealt with but the additional work in doing so is straightforward and we omit the details.

Problem 2.1. *Determine efficiently whether or not $\overline{\zeta}$ intersects Q before intersecting d_2.*

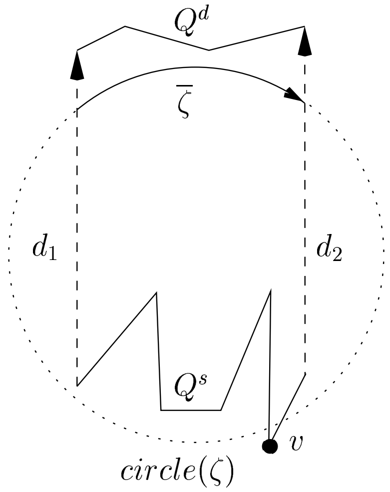
The error in [2] relates to how this problem is solved. Assume that d_1 and d_2 are directed such that, at their respective intersection with $\overline{\zeta}$, they are directed towards the exterior of circle(ζ). Accordingly, we refer to the endpoints of d_1 (d_2) as the source vertex and destination vertex of d_1 (d_2) based on this direction. Let the two chains of Q, joining d_1 and d_2, be Q^s and Q^d where Q^s joins the source vertices of d_1 and d_2 while Q^d joins the destination vertices of d_1 and d_2. See Figure 2.1.

In [2] is explained how to determine whether or not $\overline{\zeta}$ intersects Q^d. We take issue with the algorithm that detects whether or not $\overline{\zeta}$ intersects Q^s. This algorithm claims to solve the following problem:

Problem 2.2. *Determine efficiently whether or not $\overline{\zeta}$ intersects Q^s before intersecting d_2.*

We denote the radius of ζ by ζ_r. In [2], whether or not ζ intersects Q^s is determined by using the farthest point Voronoi diagram of the vertices of Q^s. Let v be a farthest vertex of Q^s from ζ_c. Then in [2] it is claimed that ζ intersects Q^s if and only if dist$(v, \zeta_c) \geq \zeta_r$ where dist(a, b) denotes the Euclidean distance between points a and b. See [2] for some optimizations

Figure 2.1: *Using the farthest point Voronoi diagram to find the farthest point from ζ_c. A vertex v is discovered and reported as being exterior to circle(ζ). However $\overline{\zeta}$ does not intersect Q^s.*



of this procedure. The optimizations are useful and can still be applied in combination with the corrected algorithm we provide.

Unfortunately the claim itself is not always valid. The difficulty is that circle(ζ) may intersect Q^s even if $\overline{\zeta}$ does not. In other words the correct claim is that circle(ζ) intersects Q^s if and only if dist$(v, \zeta_c) \geq \zeta_r$. This difficulty can lead to the algorithm in [2] falsely reporting that arc $\overline{\zeta}$ intersects Q^s as shown in Figure 2.1.

3 The Correction

To solve the circular ray shooting problem we use the algorithm in [2] except that we replace the sub-algorithm to solve Problem 2.2 with a new algorithm. In order to make this replacement we also need to modify $\mathcal{H}$ by replacing one data structure with another for each subpolygon contained in $\mathcal{H}$. We denote the version of $\mathcal{H}$ with the new data structure by $\overline{\mathcal{H}}$. The new data structure is also the source of the additional costs associated with our algorithm.

For any point or vertex p we denote by p_y the y-coordinate of p. We make the following observation:

Observation 3.1. *Arc $\overline{\zeta}$ intersects Q^s if and only if there exists a vertex v in $V(Q^s)$ with y-coordinate v_y such that dist$(v, \zeta_c) \geq \zeta_r$ and $v_y \geq \zeta_y$.*

We point out that, in our algorithm, we use ζ_c when we need a value interior to circle(ζ); we need such a value to indicate that no value on or exterior to circle(ζ) and above ζ_y has yet been found. We denote the vertices that determine a farthest point Voronoi diagram $\mathbb{V}$ by $V(\mathbb{V})$.

In order to solve Problem 2.2 we construct the farthest point Voronoi diagram of Q^s as is done in [2]. However, we construct it using divide and conquer by recursively splitting the vertices of Q^s into two halves with one half above the other. Furthermore, we keep all the Voronoi diagrams so constructed in a balanced tree structure. This tree can be constructed efficiently as follows:

Let m be the number of vertices in Q^s. We denote these Voronoi diagrams by the set $\{\mathbb{V}_k : 1 \le k < 2m\}$ where, for $1 \le j < m$, $\mathbb{V}_{2j}$ and $\mathbb{V}_{2j+1}$ are the two components that constitute Voronoi diagram $\mathbb{V}_j$ and furthermore the vertices in $V(\mathbb{V}_{2j+1})$ are above the vertices in $V(\mathbb{V}_{2j})$. Now our tree structure is just an array with a pointer to $\mathbb{V}_k$, $1 \le k < 2m$ stored in position k of the array. The parent of $\mathbb{V}_l$, $2 \le l < 2m$ is $\mathbb{V}_{\lfloor l/2 \rfloor}$.

For Voronoi diagram $\mathbb{V}_j$, $1 \le j < m$, we denote by $\mathrm{mid}(\mathbb{V}_j)$ the y-coordinate midway between the maximum y-coordinate of the vertices in $V(\mathbb{V}_{2j})$ and the minimum y-coordinate of the vertices in $V(\mathbb{V}_{2j+1})$.

For Voronoi diagram $\mathbb{V}_l$, $m \le l < 2m$, we denote by $\mathrm{mid}(\mathbb{V}_l)$ the y-coordinate of the only vertex in $V(\mathbb{V}_l)$.

For $1 \le k < 2m$, if any vertices in $V(\mathbb{V}_k)$ have y-coordinate $\mathrm{mid}(\mathbb{V}_k)$ then $\mathrm{minMid}(\mathbb{V}_k)$ denotes the vertex, of these vertices, with minimum x-coordinate and $\mathrm{maxMid}(\mathbb{V}_k)$ denotes the vertex, of these vertices, with maximum x-coordinate. Otherwise $\mathrm{minMid}(\mathbb{V}_k) = \mathrm{maxMid}(\mathbb{V}_k) = \zeta_c$.

For Voronoi diagrams $\mathbb{V}_k$, $1 \le k < 2m$, we denote by $\mathrm{far}(\mathbb{V}_k, p)$ a vertex of $V(\mathbb{V}_k)$ that is the farthest from point p. We further denote by $\mathrm{farAbove}(\mathbb{V}_k, p)$ a vertex that is the farthest from point p from the subset of the vertices of $V(\mathbb{V}_k)$ whose y-coordinates are above or equal to p_y; if there is no such vertex then $\mathrm{farAbove}(\mathbb{V}_k, p)$ denotes p.

Our algorithm is based on the following observation:

Observation 3.2. *Let* $q = \mathit{farAbove}(\mathbb{V}_k, \zeta_c)$ *where* $1 \le k < 2m$. *Then*

`If` $\mathrm{mid}(\mathbb{V}_k) = \zeta_y$ `then`
  `if` $k < m$ `then`
    $q = \mathrm{far}(\{\mathrm{far}(\mathbb{V}_{2k+1}, \zeta_c),$
      $\mathrm{minMid}(\mathbb{V}_k), \mathrm{maxMid}(\mathbb{V}_k)\}, \zeta_c)$
  `else` $q = \mathrm{minMid}(\mathbb{V}_k)$.
`If` $\mathrm{mid}(\mathbb{V}_k) < \zeta_y$ `then`
  `if` $k < m$ `then` $q = \mathrm{farAbove}(\mathbb{V}_{2k+1}, \zeta_c)$
  `else` $q = \zeta_c$.
`If` $\mathrm{mid}(\mathbb{V}_k) > \zeta_y$ `then`
  `if` $k < m$ `then`
    $q = \mathrm{far}(\{\mathrm{far}(\mathbb{V}_{2k+1}, \zeta_c),$
      $\mathrm{farAbove}(\mathbb{V}_{2k}, \zeta_c)\}, \zeta_c)$
  `else` $q = \mathrm{minMid}(\mathbb{V}_k)$.

From Observation 3.2 it is clear that we can compute $\mathrm{farAbove}(\mathbb{V}_1, \zeta_c)$ using at most $O(\log m)$ farthest point Voronoi diagram searches. Thus we can solve Problem 2.2 in at most $O(\log^2 m)$ time. The hierarchy of farthest point Voronoi diagrams that we need requires $O(m \log m)$ time and space to construct. Furthermore a hierarchy of farthest point Voronoi diagrams must be constructed for each polygon in $\overline{\mathcal{H}}$ so in total we need $O(n \log^2 n)$ time and $O(n \log^2 n)$ space to construct the necessary data structure. A circular ray shooting query now requires $O(\log^3 n)$ time.

4 Can We Build a Better Voronoi Diagram?

The algorithm from the previous section was based on the farthest point Voronoi diagram. However we can attack Problem 2.2 using a different Voronoi diagram.

For two arbitrary points a, b on the plane we denote the distance measure $\mathrm{dist}_y(a, b) = \mathrm{dist}(a, b)$ if $a_y \le b_y$ and $\mathrm{dist}_y(a, b) = 0$ otherwise. Note that this distance measure is not a metric since $\mathrm{dist}(a, b) = \mathrm{dist}(b, a)$ only if $a_y = b_y$. We call the Voronoi diagram based on this distance measure the *asymmetric farthest point Voronoi diagram* which we shorten to the *AFP Voronoi diagram*. For a query point q we refer to the site in the AFP Voronoi diagram associated with the region containing q as the asymmetric farthest site from q.

From the results in the previous section it is clear that we can represent the AFP Voronoi diagram of a set of m points in the plane by a data structure that requires $O(m \log m)$ time to construct, uses $O(m \log m)$ space and allows the asymmetric farthest site from an input query point to be found in $O(\log^2 m)$ time. This is more expensive than the cost of constructing and using a farthest point Voronoi diagram for the same set of points by a factor of $O(\log m)$ for both the query time and the storage required. An obvious approach for avoiding these additional costs is to construct the actual AFP Voronoi diagram of the set of points and apply standard methods for searching in Voronoi diagrams to find the asymmetric farthest site from a query point. This approach reduces the cost of a query from $O(\log^2 m)$ time to $O(\log m)$ time. Using this data structure we can reduce the cost of a circular ray shooting query from $O(\log^3 n)$ time to $O(\log^2 n)$ time.

Unfortunately the cost of constructing the AFP Voronoi diagram can be much higher than for constructing the standard farthest point Voronoi diagram. In fact the AFP Voronoi diagram may require $O(m^2)$ edges.

We show this by constructing a point set S for which the AFP Voronoi diagram has $O(|S|^2)$ edges. We denote the AFP Voronoi diagram of S by $\mathbb{V}_A(S)$. Instead of constructing $\mathbb{V}_A(S)$ we will a construct a simplification of $\mathbb{V}_A(S)$ which has fewer edges. Let T be this simplified diagram. We show that T has $O(|S|^2)$ edges.

To begin, assume S consists of two equal size subsets $S_b = \{b_1, b_2, \dots, b_m\}$ and $S_r = \{r_1, r_2, \dots, r_m\}$. For each region t of T we either associate t with a point $s \in S_b$ or we associate t with the set of points S_r. We further denote the set of regions associated with a point $s \in S_b$ by T_s. Our construction has the property that T_s contains $m-1$ regions of T for each point $s \in S_b$. Furthermore, each region t_i in T_s is contained in a region t_j of $\mathbb{V}_A(S)$ whose asymmetric farthest site is s and whose boundary contains at least as many edges as t_j. Thus T has $O(m^2)$ edges and so does $\mathbb{V}_A(S)$.

We construct T by constructing incrementally a sequence of diagrams $T_1, T_2, \dots, T_m = T$ where T_1 is the is the diagram constructed based on the point in S_b and T_i, $2 \leq i \leq m$, is the diagram constructed based on the points $S_b \cup \{r_1, r_2, \dots, r_i\}$.

We construct T_1 as follows. Let ζ be a circle and let the points of S_b be distributed in reverse order strictly between the six o'clock and nine o'clock position on ζ. Now T_1 is just the farthest point Voronoi diagram of S_b.

We construct T_2 from T_1 as follows. We place r_1 at the bottom of ζ and r_2 at the left most point on ζ. We now surrender to S_r all territory on or below the line $y = \zeta_y$. We also surrender to S_r all territory above this line and further from r_1 or r_2 than any point in S_b. The resulting diagram is T_2.

We construct T_3 from T_2 as follows. Let $y = c_1$ be a line above ζ_c and let r_3 be a point on the line $y = c_1$ and sufficiently far to the right that any point q_1 on the line is farther from one r_2 or r_3 than q_1 is from any point in S_b. We now surrender to S_r all territory that is both on or above the line $y = c_1$ and on or to the left of the vertical line containing r_3. This construction partitions each region associated with a point in S_b into two regions, one between the lines $y = \zeta$ and $y = c_1$ and one above the line $y = c_1$ and to the right of r_3. The latter region is unbounded. The resulting diagram is T_3. See Figure 5.1.

We construct T_4 from T_3 as follows. Let $y = c_2$ be a line sufficiently above the line $y = c_1$ that, for each point $b_i \in S_b$, the highest region associated with point b_i is intersected by the line $y = c_2$. Let r_4 be a point on the line $y = c_2$ and sufficiently far to the right that any point q_2 on the line is further from one of r_2 or r_4 than q_2 is from any point in S_b. We now surrender to S_r all territory that is both on or above the line $y = c_2$ and on or to the left the vertical line containing r_4. For each point $b_i \in S_b$, this construction partitions the highest region associated with b_i into two regions. One of the two regions is below the line $y = c_2$ and one is above and to the right of r_4. The latter region is unbounded. The resulting diagram is T_4.

We repeat the pattern of the above constructions until all points in S_r have been added and $T_m = T$ has been constructed. Now each point in S_b has $m-1$ regions associated with it so there are $m * (m-1) + 1$ regions in total in T. Furthermore T satisfies the requirements stated for it and so $\mathbb{V}_A(S)$ must contain $O(m^2)$ regions as well.

Although this result points out that it may be expensive to construct T, matters may not be as bad as they appear. The distribution of points we used to construct our example is contrived so for reasonably distributed sets of points it may be that the AFP Voronoi diagram of a set of points can be constructed at a reasonable cost.

In the case of the circular ray shooting problem we also have the advantage that the number of vertices in the AFP Voronoi diagrams may be much smaller than the number of vertices in the polygons from which they are constructed.

Consider again the polygon Q referred to in Problem 2.1. From [2] we know that Q is an x-monotone polygon and furthermore it is a subpolygon of generally larger polygon say Q'. Thus Q may have far fewer vertices than Q'. In addition, once the optimizations described in [2] are applied, we only need to construct the farthest point Voronoi diagram (or AFP Voronoi diagram) to a subset of the vertices in Q^s to solve Problem 2.2. If the size of this subset is sufficiently smaller that the number of vertices in Q' then we may be able to afford to construct the AFP Voronoi diagram in order to solve Problem 2.2. If this is true for a sufficient number of the subpolygons in $\mathcal{H}$ then we can solve the circular ray shooting Problem in the complexities claimed in [2] for polygon P.

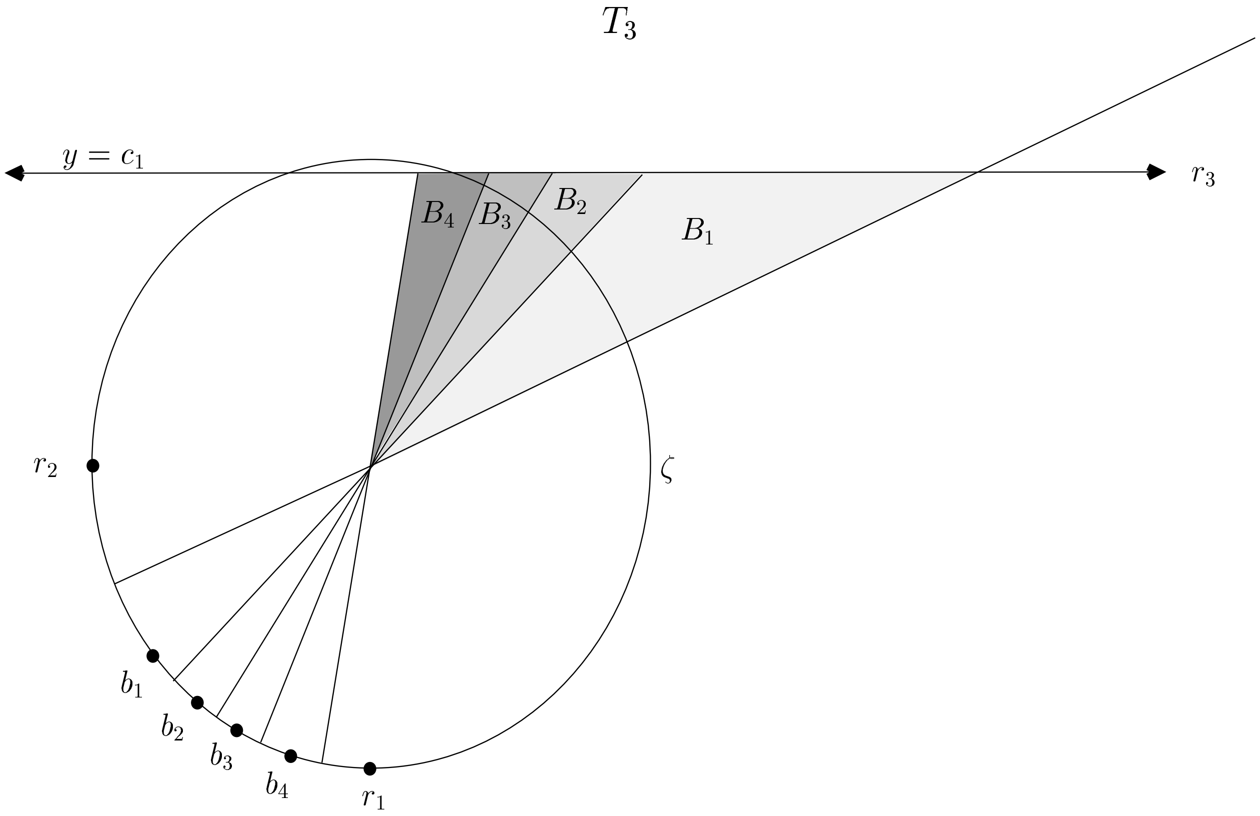
5 Conclusion

We have shown that the algorithm in [2] has an error. This error can be fixed but at the cost of an extra $O(\log n)$ factor in both the query time and the space to store the necessary data structure. We have also seen that by using a the AFP Voronoi diagram, the $O(\log n)$ factor lost in the query time can be recovered. Unfortunately, in the worst case this can increase the amount of space required by a further $O(n/\log n)$ factor. However, in some circumstances, we may be able to avoid the additional costs in using the AFP Voronoi diagram and, in those circumstances, achieve the complexity bounds originally claimed in [2] for the circular ray shooting problem.

References

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

[2] 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.



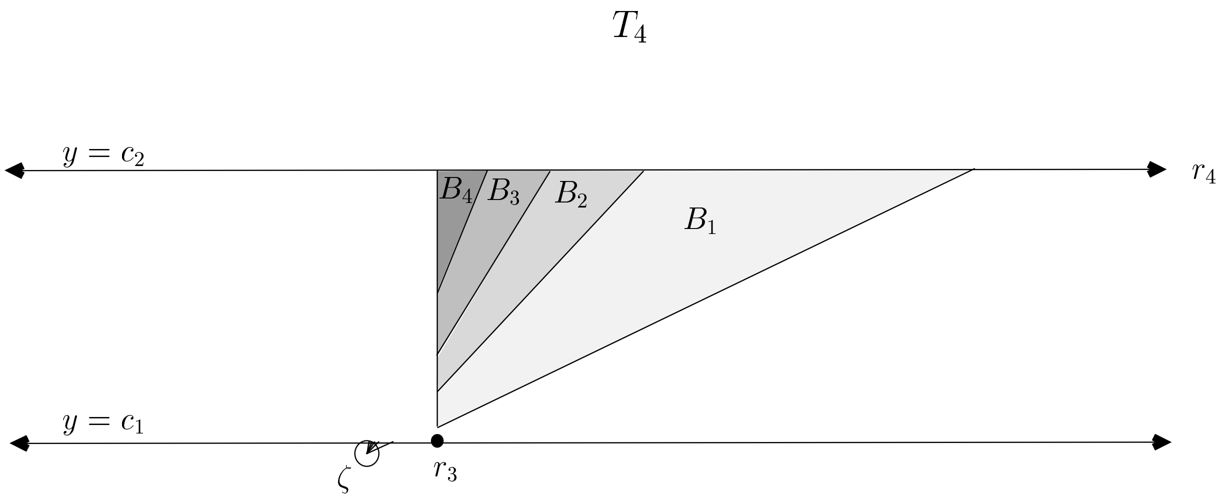


Figure 5.1:

The diagrams T_3 and T_4 when $S_b = \{b_1, b_2, b_3, b_4\}$ and $S_r = \{r_1, r_2, r_3, r_4\}$.
The regions associated with b_i are marked B_i, $1 \le i \le 4$.