# Smallest Intersecting Circle for a Set of Polygons 

Peter \begin{tabular}{ccc}
Otfried <br>
René

 

Joachim <br>
Michiel

 

Christian <br>
Antoine

 

Marc <br>
Alexander
\end{tabular} Esther

October 4, 2005

## 1 Introduction

Motivated by automated label placement of groups of islands, we consider the following problem: given a set $S=\left\{P_{1}, \ldots, P_{m}\right\}$ of $m$ polygons, with $n$ vertices in total, find the circle $c$ of smallest radius such that $c$ intersects $P_{i}$ for $1 \leq i \leq m$.

## 2 The Furthest Polygon Voronoi Diagram

The center of the optimum circle lies on an edge or vertex of the furthest polygon Voronoi diagram (FPVD). The furthest polygon Voronoi diagram is the subdivision of the plane such that within each cell, one polygon of the set is the furthest polygon (where the distance of a point to a polygon is the shortest distance); see Figure 1 for an illustration.


Figure 1: The furthest polygon Voronoi diagram.

The FPVD of a set of $m$ polygons with $n$ vertices in total is an abstract furthest site Voronoi diagram [9]. Algorithms to compute abstract furthest site Voronoi diagrams require $O(n \log n)$ time, assuming that sites and bisectors have constant complexity. In our situation, we cannot use these results directly, because bisectors can have complexity $O(n)$. However, the fact that the diagram has $O(m)$ cells does hold. By Euler's formula, this implies that the number of vertices of degree three is $O(m)$ as well. Although the bisectors can have linear complexity in $n$, we show that the number of vertices of degree two is $O(n)$.

Lemma 1 The FPVD has $O(n)$ vertices of degree two.
Proof.Just a sketch.. An FPVD-vertex $v$ of degree two lies on a bisector between two polygons, and is defined by two features of one of the polygons, and one of the features of the other. Imagine a circle $c$ that touches two features $f_{1}$ and $f_{2}$ of the polygon $P_{i}$ and whose center lies on $b\left(f_{1}, f_{2}\right)$. If we move the center of $c$ over $b\left(f_{1}, f_{2}\right)$, we grow the radius $r_{c}$. At a certain position $t_{\text {start }}$, we start intersecting all polygons $P_{j}$, for $1 \leq j \leq m$, and at another position $t_{\mathrm{end}}$, we (might) stop intersecting all polygons. In any case, the portion of $b\left(f_{1}, f_{2}\right)$ on which the circle $c_{t}$ intersects all polygons is a single connected component. Exactly the two end points of this connected component yield a degree-two vertex in the FPVD. So, all bisectors in the Voronoi diagrams of individual polygons produce at most two vertices in the FPVD. Therefore, we have that the number of degree-two vertices of the FPVD is $\sum_{1 \leq i \leq m} O\left(n_{i}\right)=O(n)$.

Since the FPVD has total complexity $O(n)$, we can construct it in an incremental way in $O(m n \log n)$ time. The algorithm is described below and is taken from [10]. We are given a set $\left\{P_{1}, P_{2}, \ldots, P_{m}\right\}$ of polygons, with $\left\{n_{1}, n_{2}, \ldots, n_{m}\right\}$ vertices, respectively, and we perform the following steps:

1. Compute a rectangle $R$ that encloses all $m$ polygons.
2. Compute the bisector of $P_{1}$ and $P_{2}$ and overlay this with the boundary of $R$; we denote this subdivision of $R$ by $S_{2}$.
3. For $i=3 . . m$ do:
(a) Compute the (closest) Voronoi diagram of $P_{i}$ and overlay it with $R$.
(b) Traverse the two overlays of $R$, one with $S_{i-1}$ and one with $\operatorname{VD}\left(P_{i}\right)$ simultaneously; update the overlay with $S_{i-1}$ such that it includes the parts where $P_{i}$ is further than all polygons $P_{j}$, for $1 \leq j \leq i$.
(c) For all points on the boundary where there is a $P_{j}$ that is equally far as $P_{i}$, overlay the bisector of $P_{i}$ and $P_{j}$ with the cell of $P_{j}$ in $S_{i-1}$. This gives us all boundaries between the cells of $P_{i}$ and $P_{j}$ in $S_{i}$. The boundary intersections give cells of of other polygons with which $P_{i}$ also has a boundary, and these are treated in the same way. When we have found all bisectors for $P_{i}$ starting at the boundary of $R$, we add the relevant bisector pieces, and we have constructed $S_{i}$ from $S_{i-1}$.
When we add polygon $P_{i}$ with $n_{i}$ vertices, we construct its Voronoi diagram in $O\left(n_{i} \log n_{i}\right)$ time [1, 11]. By Lemma 1, simultaneous traversal of the boundary of $R$ takes at most $O(n)$ time. The bisector with $P_{j}$ can be computed in $O\left(\left(n_{i}+\right.\right.$ $\left.\left.n_{j}\right) \log \left(n_{i}+n_{j}\right)\right)$ time [11, 1], and the intersection with the cell of $P_{j}$ can be done in the same amount of time, asymptotically [2,3]. Since $\sum_{1 \leq i \leq m} n_{i}=n$, the time needed to add $P_{i}$ is $O(n \log n)$, from which the $O(m n \log n)$ time bound for the construction of $S_{m}$ follows.

We have a linear number of candidates for the location of the center of the smallest intersecting circle, namely all vertices and edges of $S_{m}$. For each edge or vertex of the FPVD, we know which (two or three) polygons are furthest away, and thus we can compute in $O(1)$ time what the radius of the smallest intersecting circle with its center located at the candidate location is. Finally, we choose the candidate circle with the smallest radius.

## 3 Special Cases

### 3.1 Convex Polygons

In this section, we look at the special case in which all polygons in $S$ are convex, and develop an $O(n \log n)$ time decision problem. We then apply parametric search [8] to obtain an $O\left(n \log ^{2} n\right)$ time algorithm for the optimization problem.

In the decision problem, we are given a radius $r$, and we want to determine whether there exists a circle $c_{r}$ that intersects $P_{i}$ for $1 \leq i \leq m$. The first step is to compute the Minkovski sum of all polygons with a disk of radius $r$. If the answer to the decision problem is yes, then these blown-up polygons have a nonempty intersection. Because all polygons are convex, their Minkovski sums are convex as well, and thus we can compute their common intersection in $O(n \log n)$ time.

Peter found this paper [7], which presents an $O(n)$ time algorithm to compute the smallest radius for which there exists a disk that intersects all convex polygons.

### 3.2 Rectilinear Polygons

Now we look at the rectilinear version of our problem: all edges of the polygon are axis-parallel and the metric is the $L_{\infty}$ metric.

We first describe our decision algorithm for radius $r$. We blow up each polygon $P_{i}$ by an $L_{\infty}$ radius $r$, and obtain a polygon $P_{i}(r)$. If $n_{i}$ denotes the number of vertices of $P_{i}$, then $P_{i}(r)$ is a rectilinear polygon with at most $n_{i}$ vertices, so it can be partitioned into $O\left(n_{i}\right)$ axis-parallel rectangles. We now have a collection of $O(n)$ axis-parallel rectangles and we want to decide whether $m$ of them have a common intersection. An algorithm by Asano and Imai [6] achieves this in $O(n \log n)$ time.

Let $X=\left\{x_{1}, x_{2}, \ldots, x_{n}\right\}$ be the set of $x$-coordinates of the vertices of the input polygons, and let $Y=\left\{y_{1}, y_{2}, \ldots, y_{n}\right\}$ be the set of the $y$-coordinates. Notice that the optimal radius $r$ is equal to $x_{i}-x_{j}$ or $y_{i}-y_{j}$ for some $i, j \in$ $\{1, \ldots, n\}$. In other words, we only need to consider values of $r$ in $(X \cup Y)-$ $(X \cup Y)$. Using an algorithm by Johnson and Frederickson [4], for any integer $k$, we can find the $k$-th smallest element of $(X \cup Y)-(X \cup Y)$ in time $O(n \log n)$. As the cardinality of $(X \cup Y)-(X \cup Y)$ is $O\left(n^{2}\right)$, using our decision algorithm, we can find $r^{*}$ by binary search in time $O\left(\log \left(n^{2}\right) n \log n\right)$, which is $O\left(n \log ^{2} n\right)$ time.

## 4 Approximation Algorithm

In this section, we present an $O(n \sqrt{n})$ time 2-approximation algorithm for the problem.

Consider we are given an optimum solution $c_{o p t}$ with radius $r_{o p t}$. By definition, $c_{o p t}$ intersects all polygons in $S$, and thus all $P_{i}$ 's have at least one boundary point that lies on $c_{o p t}$. Now take any polygon $P_{i}$ and a point $p$ that lies in opt $\cap \partial P_{i}$. If we place a circle $c$ centered at $p$, with radius $2 r_{\text {opt }}$, then we have that $c \supset c_{o p t}$. Therefore, if we would find the optimum circle that intersects all $P_{j}, j \neq i$, and which has its center on the boundary of $P_{i}$, we find a circle with radius at most two times $r_{o p t}$ and thus we have a 2 -approximation.

Since the above argument holds for any polygon in $S$, we choose the smallest one, say $P_{k}$. The number of edges of $P_{k}$ is at most $n / m$. Finding the optimum over the boundary of $P_{k}$ requires that we compute the optimum over each of the $n / m$ edges. To compute the optimum on an edge $e$, we look at the distance functions of the other edges, restricted to the supporting line of $e$. For each polygon $P_{j}, j \neq i$, we compute the lower envelope of the (bivariate) distance functions of its edges on $e$, and then we compute the upper envelope of the $m$ lower envelopes. This upper envelope has linear complexity, and can be computed in $O(n \log n)$ time. Therefore, we can find a 2 -approximation in $O\left(\frac{n^{2}}{m} \log n\right)$ time. @Christian: Can you fill in the details of the last two claims?

Recall from Section 2 that we can compute the exact optimum solution in $O(m n \log n)$ time. If we run the exact and approximation algorithms in parallel, with running times $O\left(\frac{n^{2}}{m} \log n\right)$ and $O(m n \log n)$ time, and stop whenever the first algorithm terminates, we are guaranteed to have a 2 -approximation. The minimum of the two running times is $O(n \sqrt{n})$.

## 5 3sum-hardness

We show that the problem Smallest-Intersecting-Circle of finding the smallest intersecting circle for a given set of $m$ non-disjoint simple polygons is 3sum-hard, by showing that deciding whether the $m$ polygons have a point in common is 3sum-hard (Empty-Polygon-Intersection). We give a reduction from the problem Strips-Cover-Box in which we want to decide whether the union of a set of $n$ strips covers a given axis-parallel rectangle $R$ completely. This problem is known to be 3SUM-hard [5].

Let $S=\left\{S_{1}, \ldots, S_{n}\right\}$ be the set of strips, and $B$ be an axis-parallel rectangle, that are the input of Strips-Cover-Box; we transform this into an instance of Empty-Polygon-Intersection in the following way:


Figure 2: (a) The complement of $S_{i}$ within $B$ is connected. (b) The complement of $S_{i}$ within $B$ is disconnected.

- We create a set $\mathcal{P}$ of polygons. For every $\operatorname{strip} S_{i}$ in $S$, we add a polygon $P_{i}$ to $\mathcal{P}$. First, we compute the complement of $S_{i}$ within $B$. If $B \backslash S_{i}$ is connected, $P_{i}$ simply is this connected component, see Figure 2(a). If $B \backslash S_{i}$
is disconnected, we construct the simple polygon $P_{i}$ by connecting the two components by a 'handle' outside of $B$, as illustrated in Figure 2(b). Finally, we add $P_{n+1}=B$ to $\mathcal{P}$. Clearly, the sum of the number of vertices of the polygons in $\mathcal{P}$ is linear in $n$. We take $\mathcal{P}$ as the input to our problem Empty-Polygon-Intersection.
- Now it is easy to verify that the rectangle $B$ is covered by the strips in $S$, that is, $B \subseteq \bigcup_{i=1}^{n} S_{i}$, if and only the polygons in the set $\mathcal{P}$ have an empty intersection, that is, $\bigcap_{P_{i} \in \mathcal{P}} P_{i}=\emptyset$.

The reduction trivially takes $O(n)$ time, and thus we conclude that Smallest-Intersecting-Circle is 3sum-hard.

## References

[1] F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry, pages 201-290. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000.
[2] J. L. Bentley and T. A. Ottmann. Algorithms for reporting and counting geometric intersections. IEEE Trans. Comput., C-28(9):643-647, Sept. 1979.
[3] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational Geometry: Algorithms and Applications. Springer-Verlag, Berlin, Germany, 2nd edition, 2000.
[4] G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $X+Y$ and matrices with sorted rows and columns. J. Comput. Syst. Sci., 24:197-208, 1982.
[5] A. Gajentaan and M. H. Overmars. On a class of $O\left(n^{2}\right)$ problems in computational geometry. Comput. Geom. Theory Appl., 5:165-185, 1995.
[6] H. Imai and T. Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. J. Algorithms, 4:310-323, 1983.
[7] S. Jadhav, A. Mukhopadhyay, and B. K. Bhattacharya. An optimal algorithm for the intersection radius of a set of convex polygons. J. Algorithms, 20:244-267, 1996.
[8] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. J. ACM, 30(4):852-865, 1983.
[9] K. Mehlhorn, S. Meiser, and R. Rasch. Furthest site abstract Voronoi diagrams. Int. J. Comput. Geom. ©3 Appl., 11:583-616, 2001.
[10] M. van Kreveld and T. Schlechter. Automated label placement for groups of islands. The 22nd International Cartographic Conference, A Coruña, Spain, 2005.
[11] C. K. Yap. An $O(n \log n)$ algorithm for the Voronoi diagram of a set of simple curve segments. Discrete Comput. Geom., 2:365-393, 1987.

