Faster Circle Packing with Application to Nonobtuse Triangulation

David Eppstein*

Department of Information and Computer Science
University of California, Irvine, CA 92717

Tech. Report 94-33

August 9, 1994

Abstract

We show how to pack a non-simple polygon with $O(n)$ tangent circles, so that each remaining region is adjacent to at most four circles, in total time $O(n \log n)$. This improves a previous $O(n \log^2 n)$ bound. As a consequence, we can triangulate the polygon with right and acute triangles in the same bounds.

*Work supported in part by NSF grant CCR-9258355.

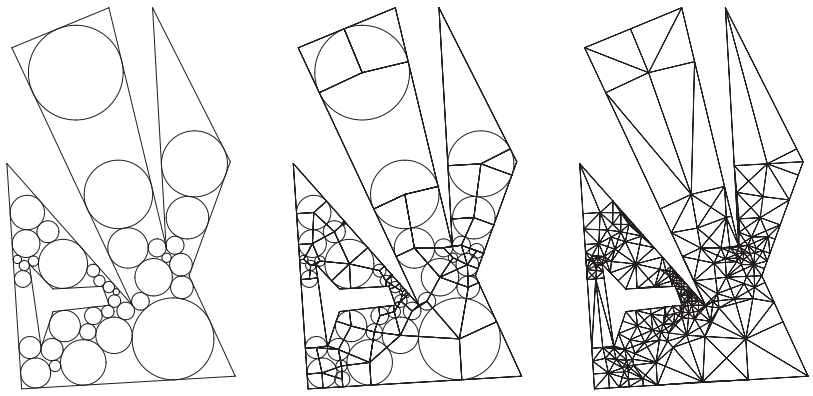

Figure 1. Non-obtuse triangulation steps: (a) pack polygon with circles; (b) connect circle centers; (c) triangulate remaining polygonal regions. (Courtesy of M. Bern)

1 Introduction

Bern et al. [2] recently described a new algorithm for triangulating any polygonal region (possibly with holes), in such a way that no triangle has an acute angle; such triangulations are useful as unstructured meshes for the finite element method. Their method only adds $O(n)$ new Steiner points, improving previous quadratic methods [1]. However the algorithm they describe takes $O(n \log^2 n)$ time. In this paper we describe a technique for performing a key step of the algorithm, that speeds the total time up to $O(n \log n)$. This is optimal as any triangulation algorithm for polygons with holes requires $\Omega(n \log n)$ time.

The algorithm of [2] is based on a technique of *circle packing*. In outline, it performs the following steps:

1. *Protect reflex vertices* of the polygon by placing circles on either side of them, small enough that they do not intersect each other or other features of the polygon.

2. *Connect holes* of the polygon by packing a collection of disjoint circles, tangent to edges of the polygon or to previously placed circles, so that the remaining area forms a collection of simply connected regions with circular-arc sides.

3. *Simplify* each region by packing it with further circles until each remaining region has three or four circular-arc sides (Figure 1(a)).

4. *Partition* the polygon into 3-, 4-, and 5-sided polygonal regions by connecting the centers of tangent circles (Figure 1(b)).

5. *Triangulate* each region with nonobtuse triangles (Figure 1(c)).

Bern et al. show how to perform steps 1 and 3-4 in time $O(n \log n)$ each; steps 5-6 can be performed in linear time. However their algorithm for step 2 takes $O(n \log^2 n)$ time, and they left as an open problem the speed-up of this step to $O(n \log n)$. It is this problem which we solve.

Our method for connecting the holes of a polygon involves combining several known techniques as building blocks. We build a *Voronoi diagram* of the polygon boundary and use it to compute a *minimum spanning tree*. The diameter circles of the MST edges connect the holes of the polygon, however they may not be disjoint. We compute the *intersection graph* of the circles using *neighborhood system* techniques of [3]. Then we shrink the circles one at a time to reduce the number of intersections. At each step we replace one of the two tangencies of a circle with a new tangency to the intersecting circle; we use the *dynamic tree* techniques of Sleator and Tarjan [5] to determine which of the two ways of performing this replacement will preserve the property that the circles form a tree connecting all holes of the polygon.

Independently, two other groups have provided alternate solutions to the same problem (M. Bern, personal communication). W. Smith has a method involving the geometry of higher dimensional cones and spheres. M. Goodrich and R. Tamassia start with the same collection of circles defined by a minimum spanning tree, but allow them to overlap; they then triangulate the regions near intersection points of two circles by adding a third small circle protecting that point. The Goodrich-Tamassia solution is perhaps closest in spirit to the rest of [2]; however it ends up adding more circles than the solution here, therefore leading to a bigger constant factor in the $O(n)$ complexity of the resulting triangulation.

2 Constructing the Minimum Spanning Tree

As a first step, we show how to construct a minimum spanning tree, with vertices corresponding to components of the polygon's boundary (holes), and edges corresponding to straight line segments connecting pairs of holes. We will use this tree to form a collection of circles, with diameters determined by the line segments of the tree.

Recall that the first step of the triangulation algorithm of [2] adds circles near reflex vertices of the input polygon. After this step, the regions remaining outside these circles are bounded by piecewise-circular curves. We define the distance between any two such curves to be the minimum length of a line segment with endpoints on each curve (we ignore the possibility that such a segment might cross other boundary components). Note that this "distance" need not satisfy the triangle inequality; nevertheless it can be thought of as an assignment of weights to the edges of a complete graph, so we can define the minimum spanning tree of the curves to be the MST in this graph.

As with any other collection of objects in the plane, we can define the *Voronoi diagram* of the collection of curves to be a partition of the plane into *cells* such that in each cell the nearest point in the collection of curves belongs to some fixed curve of the collection. The cells need not be convex, however they are connected since a straight line segment from a point to its nearest neighbor in the collection of curves must remain within a single cell. Thus the incidence graph of cells is planar and has $O(n)$ edges. The Voronoi diagram of the circular arcs making up the boundary curves can be constructed in $O(n \log n)$ time by any of several algorithms; perhaps that of [4] is best since we do not need to construct the cell boundaries in detail. The Voronoi diagram of the curves themselves can be found from the diagram of the circular arcs by joining together cells belonging to arcs of a common curve. In this same joining stage we can keep track of the shortest line segments connecting curves with adjacent Voronoi diagram cells.

As is well known, the MST of a set of points can be found from their Voronoi diagram. We next show the corresponding result for our boundary curves.

Lemma 1. *Let ST be an edge connecting curves S and T in the MST of the curves. Then the Voronoi cells for S and T are adjacent.*

Proof: Let ST correspond to line segment st, and let x be the midpoint of (s, t). Then x must be equidistant from S and T. No third curve U can be as close to x since otherwise the MST could be made shorter by replacing ST with SU or TU. Therefore x is on the boundary of the cells for S and T. □

Corollary 1. *Let D be a circle having a line segment st of the MST as its diameter. Then D does not cross or contain any of the input boundary curves.*

Proof: If it crossed a curve U, the point x above would be closer to U than to S and T. □

Corollary 2. *We can compute the MST of the curves in time $O(n \log n)$.*

Proof: We compute the Voronoi diagram, and find the $O(n)$ pairs of adjacent cells. This gives us a planar graph in which we can find the MST in linear time. □

3 Finding Overlapping Circles

A *k-neighborhood system* [6] is a collection of disks with the property that no point in the plane is interior to more than k disks. Neighborhood systems were interested to study geometric generalizations of planar separator theorems, but they have many other properties. We next show that the diameter circles of the MST edges computed in the previous section form a neighborhood system.

Lemma 2. *The diameter circles of MST edges constructed above form a 5-neighborhood system.*

Proof: Let T be the MST above, and let x be any point in the plane. Consider the MST T' of the set formed by adding x to the previous input. As with any graph MST on the insertion of an additional vertex, T' differs from T by the addition of some number i of edges incident on x and the removal of $i - 1$ edges elsewhere. As with any isolated point site in a geometric MST, the degree of x in T' is at most six (no two edges at x can form an angle smaller than 60°). By Lemma 1, the edges corresponding to all circles containing x must be among those edges removed from T to form T'; but only $i - 1 \leq 5$ edges are removed. □

We can improve this to show that the circles form a 4-neighborhood system by using a general position argument for the placement of x. However even this bound may not be tight as we are not aware of an input giving a matching lower bound on the number of overlapping MST circles.

Miller et al. [6] show that any circle in a k-neighborhood system is intersected by $O(k)$ larger circles, so there are $O(kn)$ intersections overall. We will process the circles in order from smallest to largest, shrinking each circle to remove its intersections; this result shows that we need only perform

$O(n)$ shrinking steps. However we need to know which circles are intersected by which other circles; this can be determined by an algorithm of Eppstein et al. [3] which uses a separator-based divide and conquer to find the intersection graph of a k-neighborhood system in time $O(n \log n + kn)$.

Corollary 3. *We can list the $O(n)$ intersecting pairs of MST diameter circles in time $O(n \log n)$.*

4 Shrinking Circles

At this point we have a collection of circles connecting the holes of our input polygon, however the circles intersect each other. We next describe how to shrink these circles, maintaining the property that they connect the holes, but removing the intersections one at a time.

We process the circles one at a time, in order from smallest to largest. At each point in this process, we have a collection of circles, which can be partitioned into *processed* and *unprocessed* circles. All processed circles are smaller than all unprocessed circles. No two processed circles cross each other.

To make more formal the invariant that the collection of circles connects the holes of our input polygon, we define a graph T. We form one vertex of the graph for each circle in the collection, and one vertex for each component of the boundary of the input polygon. We connect two vertices by an edge if they correspond to certain pairs of tangent circles, or if they correspond to a boundary component and a circle tangent to that component. Initially only edges of the second kind are present, and T is the tree formed by subdividing each edge of the MST.

We maintain invariant the property that T is a spanning tree on its vertex set, and that each unprocessed circle corresponds to a degree-two vertex in T. When we process a circle, T will change only at that circle, by removing certain edges and adding certain others. In order to determine how to process a vertex, we maintain a data structure for T that can handle these changes and that can answer the following queries: given a degree two vertex x and two other vertices y and z, would y and z remain connected if we deleted x? These operations can all be performed in $O(\log n)$ time using the *dynamic tree* data structure of Sleator and Tarjan [5].

To process a circle C, we consider the previously processed circles that (before they were themselves shrunk) intersected C, one circle at a time in turn. For each such smaller circle C', we first check whether, after the

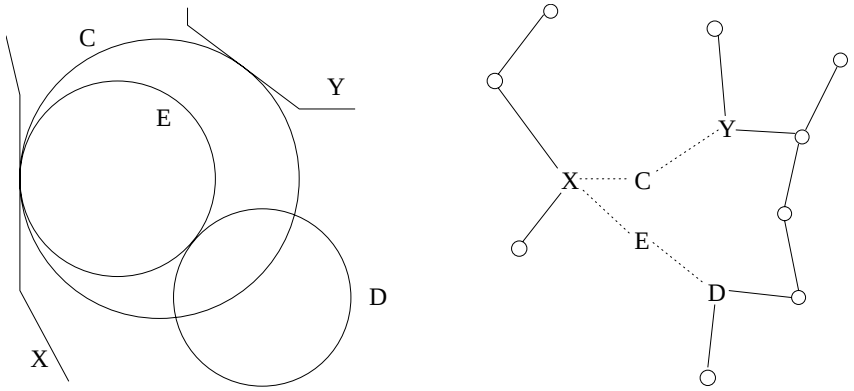

Figure 2. Circle tangent to two boundary components is crossed by another circle; replace it with smaller circle tangent to one component and to interfering circle.

changes already made to C and C', the intersection is still present. If so, let x and y be the vertices adjacent to C in T. Since C has degree two in T, its removal partitions T into two components, with x and y in separate components. C' must be in one of these components; we determine which one using the dynamic tree algorithm. Assume without loss of generality C' is in the component containing y. We then find a circle C'' contained entirely within C, tangent at the same point as before to x, and tangent also to C'. We shrink C by replacing it with C''. This process is depicted in Figure 2; in this figure, C is the largest of the three circles, and its replacement C'' is the circle entirely contained in C. Since the new circle is contained in the old one, no new circle intersections are formed, and the intersection with C' is removed. The tree T changes by the removal of edge Cy and the insertion of edge CC'; this preserves the invariants above.

After repeating this process for each intersecting pair of circles, all intersections are removed and T remains a tree. Therefore, we have found a collection of disjoint circles connecting the holes of the input polygon.

Lemma 3. *From the collection of MST diameter circles constructed earlier, we can perform the shrinking process described above in time $O(n \log n)$.*

Proof: As shown earlier, there are $O(n)$ intersections to remove. Each removal takes time $O(\log n)$ using the dynamic tree data structure [5]. □

We summarize our results.

Theorem 1. *We can add disjoint circles to any polygon with holes, or more general region with circular-arc boundaries, so that each remaining region has a connected boundary, in time $O(n \log n)$.*

By combining this with the other methods of Bern et al. [2] we can show:

Theorem 2. *We can triangulate any polygon with nonobtuse triangles, adding $O(n)$ additional vertices, in time $O(n \log n)$.*

References

[1] M. Bern and D. Eppstein. Polynomial-size nonobtuse triangulation of polygons. *Int. J. Comp. Geom. & Appl.* 2 (1992) 241–255.

[2] M. Bern, S. Mitchell, and J. Ruppert. Linear-size nonobtuse triangulation of polygons. *Proc. 10th ACM Symp. Comp. Geom.* (1994) 221–230.

[3] D. Eppstein, G. L. Miller, and S.-H. Teng. A deterministic linear time algorithm for geometric separators and its applications. *Proc. 9th ACM Symp. Comp. Geom.* (1993) 99–108.

[4] M. McAllister, D. Kirkpatrick, and J. Snoeyink. A compact piecewise-linear Voronoi diagram for convex sites in the plane. *Proc. 34th IEEE Symp. Foundations of Computer Science* (1993) 573–582.

[5] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comp. Sys. Sci.* 24 (1983) 362–381.

[6] G. L. Miller, S.-H. Teng, and S. A. Vavasis. A unified geometric approach to graph separators. *Proc. 32nd IEEE Symp. Foundations of Computer Science* (1991) 538–547.