Polynomial-Size Nonobtuse Triangulation of Polygons

Marshall Bern* David Eppstein†

Abstract

We describe methods for triangulating polygonal regions of the plane so that no triangle has a large angle. Our main result is that a polygon with n sides can be triangulated with $O(n^2)$ nonobtuse triangles. We also show that a convex polygon can be triangulated with $O(n^2)$ right triangles. Finally we show that any triangulation (without Steiner points) of a simple polygon has a refinement with $O(n^4)$ nonobtuse triangles.

1. Introduction

One of the classical motivations for problems in computational geometry has been automatic mesh generation for finite element methods. In particular, mesh generation has motivated a number of triangulation algorithms, such as finding a triangulation that minimizes the maximum angle [5]. A triangulation algorithm takes a geometric input, typically a point set or polygonal region, and produces an output that is a triangulation of the input. For a point set, this usually means partitioning the region bounded by its convex hull into triangles, such that the vertices of the triangles are exactly the input vertices. Triangulating a polygonal region usually means partitioning the region into triangles such that the vertices of the triangles are exactly the vertices of the region's boundary.: In both cases, the output must be a simplicial complex, that is, triangles intersect only at shared vertices or edges.

Mesh generation applications, however, invariably place further requirements on the triangulation, stemming from numerical analysis. Angles larger or smaller than certain bounds are frequently forbidden. To satisfy such angle bounds, it may be necessary to augment the input by adding new vertices, called *Steiner points*. Because the complexity of the finite element computation depends on the size of the mesh, the number of Steiner points should be kept small. Until recently, computational geometers neglected the more realistic Steiner versions of triangulation problems, and concentrated on problems that do not allow extra vertices.

An upper bound of 90° has special importance in mesh generation. A triangulation with maximum angle 90° is necessarily the Delaunay triangulation of its vertex set. In finite element methods for solving certain partial differential equations, a Delaunay mesh—or failing that, a nonobtuse Delaunay mesh—leads to a matrix with nice numerical propertices, such as diagonal dominance [1]. For a second, more geometric, motivation, notice that each (closed) triangle in a triangulation contains the center of its circumscribing circle exactly when all angles measure at most 90°. The "perpendicular planar dual" of such a triangulation can be formed by simply joining perpendicular bisectors of edges. Practitioners often use a mesh and its dual—sometimes with bent edges in the dual—to discretize vector fields [1]. In general, a straight-line embedding of the planar dual of a triangular mesh can be formed by placing dual vertices at the meeting point of angle bisectors [3]; however, a perpendicular planar dual simplifies the calculation of flow or forces across element boundaries.

1.1. Summary of Results

In this paper we consider the problem of triangulating a polygonal region so that all angles measure at most 90°. We give the first algorithm that produces such a nonobtuse triangulation with size (number of triangles) bounded by a polynomial in n, the number of sides of the input polygon. The only previous provably-correct algorithm, due to Baker et al. [1], has no size guarantee. Indeed, the size of the triangulation produced by the algorithm of Baker et al. depends on the geometry of the input, and hence can be arbitrarily large, even when n is fixed.

Our main result is that an arbitrary polygon can be triangulated with $O(n^2)$ nonobtuse triangles. The polygon need not be simple; it may contain polygonal holes. We also describe a related algorithm for triangulating a

*Xerox Palo Alto Research Center, 3333 Coyote Hill Rd., Palo Alto, CA 94304

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

convex polygon with $O(n^2)$ right triangles. Finally, we consider the problem of refining a given triangulation of a polygon (without Steiner points) into a nonobtuse triangulation. We achieve $O(n^4)$ triangles for arbitrary simple polygons, and $O(n^2)$ for the special case of a polygon with a "Hamiltonian" triangulation, that is, a triangulation whose dual is a path. All our algorithms can be made to run in time $O(n \log n + k)$, where k is the size of the output.

Our angle bound is the best possible for polynomial-size triangulations. Our work with John Gilbert [2] shows that any smaller bound on the largest angle would require the number of triangles to depend not only on the size of the input, but also on its *aspect ratio*.

1.2. Related Work

The original computational geometry result motivated by mesh generation is that the Delaunay triangulation of a point set maximizes the minimum angle [7, 10, 11, 14]. (It is a curiously similar fact that, for points in convex position, the farthest point Delaunay triangulation minimizes the minimum angle [6].) For polygons the minimum angle is maximized by the constrained Delaunay triangulation [9]. It is also known how to compute the triangulation that minimizes the maximum angle [5] for both point sets and polygons.

Theoretical work on Steiner triangulation problems is less common. In an earlier paper [2], we developed algorithms based on quadtrees for a number of Steiner triangulation problems. In a certain strong sense, we solved the problems of Steiner-triangulating point sets and polygonal regions with no small angles (when the minimum angle bound is sufficiently small). For each input our algorithms produce an output that has size within a constant factor of the optimal size. The number of triangles required necessarily depends not only on the size of the input, but also on its aspect ratio. We also showed how to triangulate point sets with no large angles. If the angle bound is less than 90°, the solution is essentially identical to that for no small angles; but if the only requirement is that there be no obtuse angles, the point set can be triangulated with only linearly many Steiner points.

Gerver [8] showed how to compute a "dissection" of a polygon (that is, vertices embedded within sides of triangles are allowed) with no angles larger than 72°, assuming all interior angles of the input measure at least 36°. As mentioned above, Baker et al. [1] gave a nonpolynomial algorithm for nonobtuse triangulation, thereby showing that every polygon has such a triangulation. Their algorithm also avoids small angles, and is thus inherently nonpolynomial. (For technical reasons, Baker et al. assumed the input polygon has integer coordinates, but stated that this restriction could be lifted.) Our earlier result that point sets can be triangulated with $O(n)$ nonobtuse triangles led us to suspect that the Baker et al. result could be improved, and that polygons too could be triangulated with a polynomial number of nonobtuse triangles.

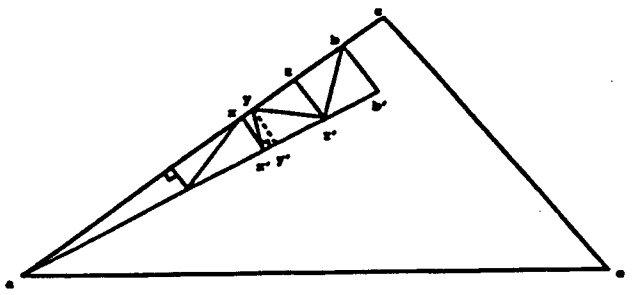

Figure 1. Side merger.

2. Obtuse with subdivided legs

In this section, we develop the main subroutine of our algorithm for general polygons. In the next section, we shall present the full algorithm.

Throughout this paper, we use *hypotenuse* to mean the longest side of a triangle and *legs* to mean the other two sides. A *subdivision point* is a vertex of a polygon at which the angle measures 180°. A *nonobtuse* (*nonacute*) angle is one measuring at most (respectively, at least) 90°. The measure of angle $\angle abc$ is denoted $|\angle abc|$.

Suppose we are given a triangle ace with hypotenuse ae as its horizontal base, with $|\angle ace| \geq 90°$, and with subdivided legs. In this section we show how to triangulate this input with nonobtuse triangles, without adding any new subdivision points to the legs. Our overall strategy is to replace the legs by ones sloped slightly closer to the horizontal, while keeping the base fixed. The region between the old legs and the new legs is triangulated in such a way as to reduce the total number of subdivision points by one.

We illustrate one of our reduction techniques in Figure 1. Assume that there are at least three subdivision points on side ac, and let b be the one adjacent to apex c. Let $|\angle ace| > 90°$ and let b' be a point in triangle ace, somewhere on the line extending from b perpendicular to ac. Project all subdivision points along ab perpendicularly away from ab onto ab', adding edges between each point and its projection. This step divides triangle abb' into one right triangle and at least two trapezoids. A trapezoid is *good* if adding one of its

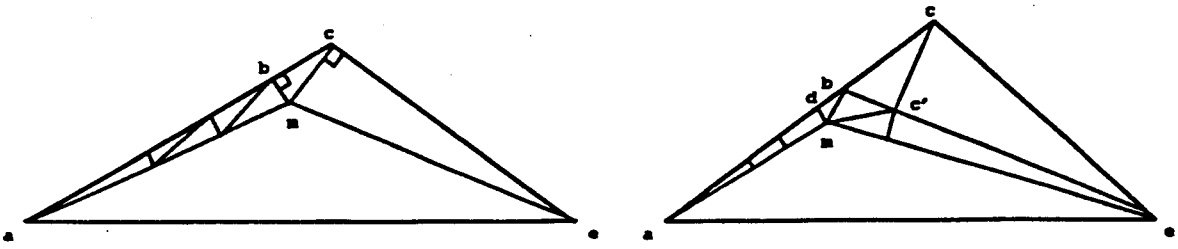

Figure 2. (a) Apex merger. (b) Delayed apex merger.

two diagonals divides it into two nonobtuse triangles. A key observation is that all trapezoids will be good if $\angle bab'$ is sufficiently small.

Now imagine moving b', starting from b, along the perpendicular to ac. There will be a last position for b' at which all trapezoids are good. Fix b' at that position, and let trapezoid $x'xyy'$ be one that would become bad if b' were moved further, as shown in Figure 1. This means that $|\angle yx'y'| = 90°$. Further assume that $x'xyy'$ is not the last trapezoid along ab, that is, $y \neq b$. Then we *merge* trapezoid $x'xyy'$ with the next trapezoid above it. This merger is accomplished by removing point y' and edge yy' to form a quadrilateral $x'xzz'$ with the one subdivision point y. This face can be triangulated with three right triangles by adding edges $x'y$ and yz'.

Merger moves such as the one just described lie at the heart of our algorithm. Our first two lemmas give the basis of an induction. Lemmas 3 and 4 give "coroutines" for the inductive step; each of these lemmas calls the other on a triangle with at most $n-1$ subdivision points.

Lemma 1. *If ace is a right triangle with one subdivision point on a leg, ace can be triangulated into three right triangles, with one subdivision point on base ae.* ∎

Lemma 2. *If triangle ace is obtuse with no subdivision points, then it can be triangulated into two nonobtuse triangles, adding one subdivision to base ae. If obtuse ace has one subdivision point on a leg, ace can be triangulated into at most five nonobtuse triangles, adding at most two subdivisions to base ae.*

Proof: If ace has no subdivision points, then we drop an altitude from c to ae and we are done. Assume that ace has one subdivision point b on leg ac. Extend a perpendicular to ac at b and a perpendicular to ce at c. If these meet inside ace, then add their meeting point m and edges bm and cm. Triangle ame has no subdivision points. If the perpendiculars meet outside of ace, then let b' be the point at which the perpendicular at b crosses ae. Add bb' and $b'c$; the remaining obtuse triangle $b'ce$ has no subdivision points. ∎

Lemma 3. *Assume triangle ace has $n \geq 2$ subdivision points, all on one leg. Using $O(n)$ nonobtuse triangles, we can reduce the untriangulated portion of ace to a nonacute triangle with $n-1$ subdivision points.*

Proof: Without loss of generality, assume ac is the subdivided leg, and let b be the subdivision point closest to c. If $|\angle ace| = 90°$, we simply add edge be and reduce to the case of an obtuse triangle with $n-1$ subdivision points. So assume $|\angle ace| > 90°$, and consider extending a perpendicular to ac at b and a perpendicular to ce at c. Assume these meet at point m in ace. If $\angle bam$ is sufficiently small, then all subdivision points along ac can be projected onto am along perpendiculars to ac, forming only good trapezoids. See Figure 2(a). We triangulate these trapezoids, add edge em, and are done. This move, in which we reduce the number of subdivision points by combining the apex and an adjacent subdivision point using the meeting point of perpendiculars, is called an *apex merger*.

So assume meeting point m does not work. Then find the last point b' in triangle ace along the perpendicular to ac at b, such that the projection forms only good trapezoids. Notice that $|\angle b'ce| > 90°$ since the apex merger did not succeed.

If b' lies on base ae, then we add edges bb' and $b'c$, triangulate all the good trapezoids, and drop an altitude from c. Now there is no untriangulated portion of ace, and the number of Steiner points on base ae is at most $n+1$.

So assume b' lies interior to ace. By our choice of b', either some trapezoid along ab can be merged with the one above it, or the last trapezoid—containing b and b'—is the one whose diagonal forms a right angle. In the first case, we perform a merger as shown in Figure 1 (called a *side merger*), add edges $b'c$ and $b'e$, and project c onto $b'e$ by dropping an edge perpendicular to $b'e$. The resulting triangle $ab'e$ has $n-1$ subdivisions on two legs.

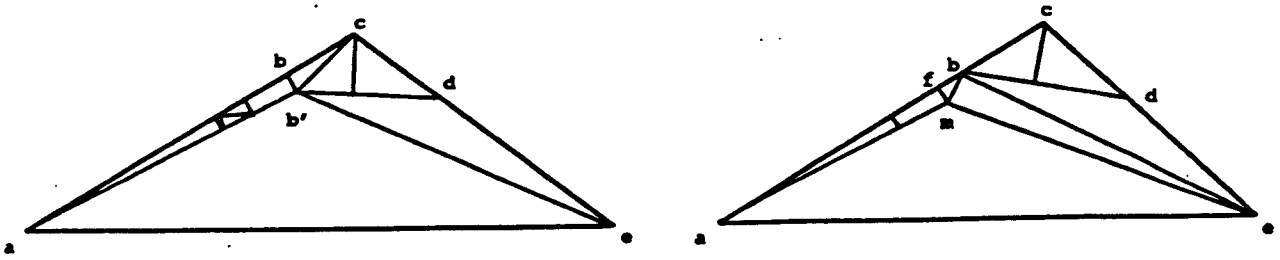

Figure 3. (a) Triangle $b'de$ is triangulated recursively. (b) Triangles bde and mbe are triangulated recursively.

In the second case, shown in Figure 2(b), we add edge be and project c onto be as above, producing point c'. We now perform an apex merger of b and d using meeting point m in the new triangle abe, and project c' perpendicularly onto me. Notice that triangle mbe has been split into three right triangles. We have again reduced to the case of $n-1$ subdivisions on two legs, handled by Lemma 4 below. ∎

Observe that the method given above either merges a pair of points without adding anything to the base, or completely finishes off all the subdivisions on ac. The same will be true of our method for the case of subdivisions on both legs. Therefore we assert that our method produces at most $n+1$ subdivision points on base ac, and at most n if ace is right. Because each reduction uses $O(n)$ triangles, the total number used will be $O(n(n-n_b+2))$, where n_b is the number of subdivisions added to the base. We use these bounds recursively in the analysis of the general inductive step below.

Lemma 4. *Assume triangle ace has $n \geq 2$ subdivision points, with at least one on each leg. Using $O(n(n-n_o))$ nonobtuse triangles, we can reduce the untriangulated portion of ace to a nonacute triangle with $n_o \leq n-1$ subdivision points.*

Proof: Let b and d be the closest subdivision points to c on ac and ce, respectively. Let n_r represent the number of subdivision points on the right leg ce.

We first try each of the two possible apex mergers. Say that the perpendiculars to ac at b and to ce at c meet at point m in ace, such that all trapezoids along ac are good. In this case, we add edges md and me. We then triangulate mde recursively, outputting at most n_r subdivisions on base me. Thus we have reduced the problem to triangulating ame, an obtuse triangle with one fewer subdivisions.

So assume no apex merger is possible, and let b' be the last point in triangle ace along the perpendicular to ac at b such that the projection from ab onto ab' forms only good trapezoids. Since no apex merger is possible, $\angle b'ce$ and $\angle b'de$ are both obtuse.

Now if b' lies on base ac, then we add edges bb', $b'c$, and $b'd$; drop an altitude from c to $b'd$; and triangulate the trapezoids along ab. This reduces the problem to triangulating $b'de$, an obtuse triangle with at most $n-1$ subdivisions.

If ac has only one subdivision point, then either an apex merger of b and c is possible, or b' lies on the base. So assume ac contains at least two subdivision points, that is, $n_r \leq n-2$, and b' lies interior to ace. By our choice of b', either some trapezoid along ab can be merged with the one above it, or the last trapezoid is the one whose diagonal forms a right angle.

In the first case, we can perform a side merger and reduce the number of subdivision points. We add edges $b'c$, $b'd$, $b'e$, and an altitude from c onto $b'd$, as shown in Figure 3(a). Triangle $b'de$ has n_r subdivisions. Triangle $b'de$ is triangulated recursively with $O(n_r(n_r-n_b+2))$ triangles, outputting n_b subdivisions on its base $b'e$. Then triangle $ab'e$ will have $n_b \leq n_r+1$ subdivisions on leg $b'e$ and $n-n_r-2$ on leg ab' (one fewer due to the merger and another fewer because b' is the apex), so we are done.

In the second case, we cannot do a side merger because the next subdivision point f below b is too close to b. As in Lemma 3, however, we can do a "delayed apex merger". We add edges bd, be, and drop an altitude from c to bd. See Figure 3(b). We perform an apex merger of f and b at meeting point m in triangle abe. We then triangulate bde recursively, outputting $n_b \leq n_r+1 \leq n-1$ subdivisions on be. Right triangle mbe is handled by Lemma 3, outputting at most n_b subdivisions on me. We have now reduced the problem to triangulating ame, a triangle with $n-n_r-2$ subdivisions on its left leg and at most n_r+1 on its right leg. Furthermore, we used only $O(n)$ triangles for each reduction in the number of subdivisions. ∎

Theorem 1. *An obtuse or right triangle with n subdivisions on its legs can be triangulated with $O(n^2)$ nonobtuse triangles, without adding any new subdivisions on the legs.* ∎

3. Arbitrary Polygons

We now show how the method of the previous section can be applied to an arbitrary polygon, possibly with holes. First we partition the polygon into some simple shapes, by cutting it with horizontal and vertical lines. Each such shape can be either triangulated directly, or can be divided into obtuse triangles (with subdivided legs) which can then be further triangulated as before.

Draw a vertical line segment through each vertex of the polygon, extending to the boundaries of the polygon. These lines divide the polygons into *slabs*, which we define as quadrilaterals with two vertical sides, possibly with subdivision points on the vertical sides.

Each vertex of a slab will be either an original input vertex, or a point where a vertical line touches the polygon boundary. Draw a horizontal line segment through each slab vertex, and extend the line segment to the last possible vertical segment. In other words, each endpoint of a horizontal segment should lie either on a vertical segment, or on the vertex inducing the horizontal, and each horizontal segment should be as long as possible with this property. A polygon so divided is shown in Figure 4.

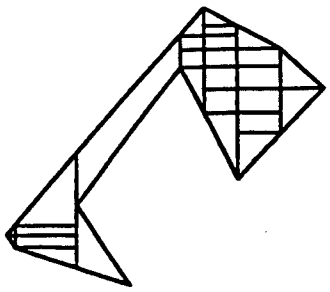

Figure 4. Polygon cut by horizontal and vertical lines.

Lemma 5. *The above steps partition the polygon into regions of four types: (1) rectangles with unsubdivided sides; (2) right triangles with hypotenuse on the boundary of the polygon and vertical leg possibly subdivided; (3) obtuse triangles with two sides on the boundary of the polygon, and one leg vertical and possibly subdivided; (4) slabs with two sides on the boundary of the polygon, and two possibly-subdivided vertical sides, that cannot be simultaneously crossed by a horizontal line.*

Proof: Any other shape must have a vertex from which a line segment can be extended vertically or horizontally. The construction only creates vertical and horizontal line segments, so any other segments must be on the polygon boundary. ∎

Theorem 2. *Any n-vertex polygon (with holes) can be triangulated with $O(n^2)$ nonobtuse triangles.*

Proof: We apply the procedure above to subdivide the polygon into the shapes listed in Lemma 5. The rectangles are easy. The right and obtuse triangles can be triangulated by the method of the previous section. The slabs can be divided by a diagonal into two obtuse triangles, that can then be triangulated by the method of the previous section.

We create $O(n)$ horizontal and vertical line segments, so there are $O(n^2)$ rectangles. Each other shape contains a portion of the polygon boundary between vertical line segments, so there are $O(n)$ such regions. There are $O(n)$ subdivisions in all these regions, arising from the horizontal line segments. Therefore the method of the previous section, when applied to the nonrectangular regions, creates $O(n^2)$ triangles. ∎

4. Convex Polygons

We now show how a convex polygon can be triangulated with all right triangles. We start out by partitioning the polygon vertically and horizontally as in the general case. However there is an important additional requirement. The choice of horizontal and vertical axes is made so that the longest diagonal of the polygon (called the *main diagonal*) is oriented horizontally. Thus this diagonal will appear in its entirety as one of the horizontal segments of the partition. We assume that the main diagonal is not part of the boundary of the polygon; otherwise the polygon can be divided into right triangles more simply by orienting the partition so that the main diagonal is vertical.

Our algorithm, explained below, periodically *extends* the partition by adding a new vertical line from boundary to boundary of the polygon. When we add a new vertical line, we also add horizontals from its endpoints to the last verticals as before, and lengthen each other horizontal segment for which the new vertical is now the last vertical. We extend the partition only $O(n)$ times; thus we end up with $O(n^2)$ rectangles and $O(n)$ subdivided right triangles. Our method for removing subdivisions in these triangles is based on Lemma 1, the fact that a right triangle with just one subdivision point on a leg can be triangulated by adding a subdivision to the hypotenuse. Our strategy is to carefully extend the partition until each triangle has at most one subdivision point.

Consider, without loss of generality, the chain of triangles extending up and to the right from the left endpoint of the main diagonal. For each subdivision point

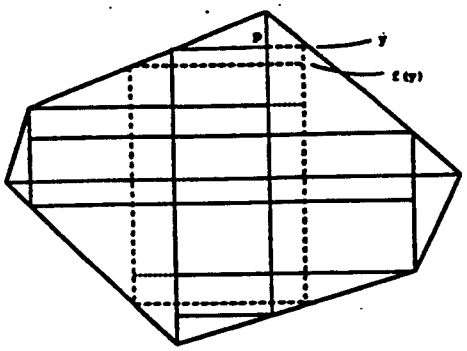

Figure 5. Convex polygon with path from subdivision point.

on one of these triangles, we "bounce" a *path* as follows. First draw a horizontal segment from the point to the polygon's boundary, and extend the subdivision by drawing a vertical line where the horizontal meets the boundary. This eliminates the original subdivision point, but creates a new one below the main diagonal. Now repeat the process to move the subdivision back above the main diagonal. Figure 5 depicts a convex polygon with its partition extended by a path.

The new subdivision point may fall on the chain extending up and to the left of the right endpoint of the main diagonal, instead of on the original chain. However it can be shown that, if this happens, the reverse cannot be the case: paths from the up-left chain cannot end on the up-right chain. If we first process the chain that can send points to the other chain, there will be no problem. So we now ignore points sent to the opposite chain.

For a subdivision point p at height y from the main diagonal, define $f(y)$ to be the height of the new subdivision point that would be created by bouncing a path from p. This function has the following properties.

Lemma 6. *The function $f(y)$ is continuous, monotone, and piecewise linear with $O(n)$ breakpoints.*

Proof: By construction, f is the composition of four such functions, corresponding to the four chains among which the path bounces. ∎

As a consequence, $f(y) - y$ is also continuous and piecewise linear. We partition the plane into horizontal strips so that within each strip $f(y) - y$ has the same sign (positive, negative, or zero). We extend the partition by bouncing paths at points with heights at which $f(y) - y$ changes sign; these paths form rectangles and therefore do not introduce new subdivisions; because there are $O(n)$ breakpoints, there are $O(n)$ strips and therefore $O(n)$ rectangles drawn in this stage. The rectangles ensure that no triangle of the partition contains portions of more than one strip. Because of the following fact, we may consider each strip independently.

Lemma 7. *For each y, $f(y)$ is in the same strip as y.* ∎

Now consider a triangle with more than one subdivision point in a strip where $f(y) > y$, and let p and q be the lowest two points in this triangle, with p lower than q. Bouncing the path from q creates a new subdivision point higher than q, and cuts off a triangle in which p is the lone subdivision point. We can repeat this process until all subdivision points in the strip are alone. Each extension creates a new lone subdivision point; therefore, after $O(n)$ extensions all subdivision points are alone in their triangles. In strips where $f(y) < y$, the process is similar, beginning with the highest two points that are not alone. Finally, in a strip where $f(y) = y$, bouncing a path from a subdivision point creates a rectangle. So in this case, all subdivision points can be immediately removed.

At this point, every remaining subdivision point is alone in its triangle. Applying Lemma 1 to the triangles, and adding diagonals to the internal rectangles, gives our result:

Theorem 3. *Any convex polygon with n vertices can be triangulated with $O(n^2)$ right triangles.* ∎

5. Refining a Given Triangulation

In this section we consider the problem of refining a given triangulation (without interior Steiner points) of a simple polygon into a nonobtuse triangulation. The edges of the input triangulation must appear in the output, possibly subdivided. This problem is a special case of—and perhaps a first step towards solving—the more general problem of refining an arbitrary straight-line planar graph with nonobtuse triangles. The more general problem has applications to computing a mesh on a polyhedral surface, and to learning polygons from examples given by a "helpful teacher". Salzberg et al. [13] have shown that, if the inside and outside of a polygon can be simultaneously triangulated without obtuse angles (allowing infinite nonobtuse wedges and right-angled strips), then a teacher who picks examples as advantageously as possible can teach a polygon to a nearest-neighbor classifier, using a number of examples proportional to the size of the triangulation.

Our strategy for this problem is to again cut the input into cases that we can handle: rectangles and obtuse triangles with subdivided legs. The cutting procedure, however, is more complicated. We process triangles in

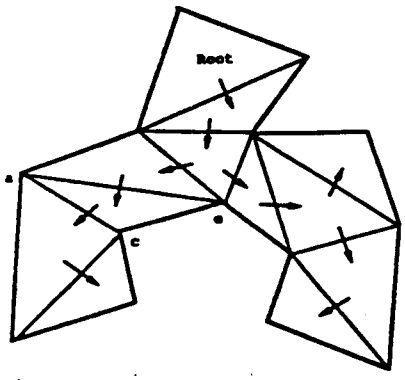

Figure 6. Triangle *ace* is backwards obtuse.

a preorder traversal of the tree that is the planar dual of the initial triangulation. Each triangle after the first one then has one "inbound" edge (the one shared with a triangle earlier in the order) and two "outbound" edges. Ideally each triangle is obtuse with its hypotenuse an outbound edge. Our preprocessing step dices up "backwards" obtuse triangles, propagating $O(n^2)$ subdivision points up the tree, in order to achieve this ideal situation. The ideal situation can be triangulated with quadratic increase in complexity.

Lemma 8. *If ace is a nonobtuse triangle with subdivisions on only one side, then ace can be triangulated into nonobtuse triangles, adding new subdivisions only to the other two sides.*

Proof: Let *ac* be the side with subdivisions. Consider the perpendicular projection e' of e onto *ac*. We add edges between e and the subdivision points adjacent to e' (though we do not add e' or edges to e'). This divides *ace* into one nonobtuse triangle without subdivisions and either one or two nonacute triangles that can be triangulated by the method of Section 2. ∎

Theorem 4. *Any triangulation of a polygon (without holes) with n vertices can be refined into a nonobtuse triangulation with $O(n^4)$ triangles.*

Proof: Let T be the tree that has a vertex for each triangle of the input and an edge between each pair of triangles that share a side. Root T at any vertex corresponding to a triangle with an *exterior* side, that is, one lying along the boundary of the polygon. Each triangle will have its sides labeled as *inbound* or *outbound*. The side that the triangle shares with its parent in T is labeled inbound; the other two sides are labeled outbound (whether exterior or shared with a child triangle).

If at this point each triangle is nonobtuse or is obtuse with its hypotenuse labeled outbound, then we can refine the triangulation as follows. We split the root triangle into two right triangles by dropping an altitude. Each subsequent triangle in the tree starts with some number of subdivision points on its inbound side, and we use Lemma 8 and the procedure of Section 2 to triangulate each triangle, adding new subdivisions only to outbound sides.

So assume that there is at least one *backwards* obtuse triangle with its hypotenuse labeled inbound. See Figure 6. We first process triangles in "upward" order given by a postorder traversal of T. A backwards obtuse triangle that has no backwards descendants is handled by simply dropping an altitude onto its hypotenuse; this introduces a subdivision point to an outbound side of the parent triangle.

We now carry this through inductively, by showing how to partition a triangle, that has subdivisions on its outbound sides, into rectangles and right triangles with subdivided legs and hypotenuse that is part of an outbound side.

Assume the triangle is *ace* and its inbound side is horizontal base *ae*. If $\angle cae$ and $\angle cea$ are both acute, then we partition *ace* by dropping vertical lines from c and from all subdivision points onto *ae*. We then draw horizontal lines from each subdivision point up to the last possible vertical, as in Figure 4. This partitioning introduces no new subdivisions to the outbound sides. If one of $\angle cae$ and $\angle cea$ is right, then we partition in the same way. This introduces new subdivisions to the vertical side of *ace*; these will be corrected in the downward pass of the algorithm.

So assume that one of $\angle cae$ and $\angle cea$ is obtuse, say $\angle cea$. We draw horizontal lines from each subdivision point on *ac* and *ce* across to the other side. Now from vertex e, from each subdivision point, and from each new vertex (where a horizontal hits *ac* or *ce*), extend a vertical line, up or down, until it contacts the last possible horizontal, base *ae* counting as a horizontal. As above, this partitioning introduces new subdivisions to the outbound sides, dealt with in the next—downward—pass of the algorithm.

The downward pass of the algorithm uses Lemma 8 and the procedure of Section 2 to triangulate in an order given by a preorder traversal of T. Each original triangle is either unchanged, which implies that it is not backwards obtuse, or it has been partitioned as above into rectangles and small triangles with hypotenuse part of an original outbound edge. Subdivisions added to sides of rectangles are passed straight through, slicing the rectangle into smaller rectangles that can be triangulated at the end. The small triangles within any one original triangle are triangulated in any order, after the rectangles have been sliced.

The total number of vertices (subdivision points and apexes of obtuse triangles) to propagate downwards is $O(n^2)$. This bound follows from the fact that, in the upwards pass of the algorithm, each apex of a backwards obtuse angle adds only $O(1)$ new subdivision points per original triangle. The downwards pass of the algorithm is quadratic as before, so overall we have obtained a nonobtuse triangulation with $O(n^4)$ triangles. ∎

In the case that the dual graph of the triangulation is simply a path, then we can improve the size bound to $O(n^2)$ by exploiting the fact that each triangle has an exterior side. Incidentally, this condition defines an interesting class of polygons: call a polygon *Hamiltonian* if it has a Steiner triangulation whose dual is a path. Equivalently, a Hamiltonian polygon is one that can be swept by an extensible line segment with endpoints on its boundary, without retracing any area. Hamiltonian polygons simultaneously generalize monotone polygons and spiral polygons (polygons with only one chain of reflex vertices); they can be recognized in time $O(n^2)$ by dynamic programming.

Theorem 5. *Assume we are given a triangulation of a polygon with n vertices, such that the planar dual of the triangulation is a path. Then this triangulation can be refined into a nonobtuse triangulation with $O(n^2)$ triangles.*

Proof: Assume we are at a generic step of the upwards pass, in original triangle ace. Assume outbound side ac is subdivided, ce is exterior, and horizontal base ae is inbound. If $|\angle cae| \geq 90°$, then the procedure of Section 2 will triangulate ace in the downwards pass of the algorithm, adding new subdivisions only to exterior side ce; thus, the upwards pass of the algorithm need not partition ace. If $|\angle ace| \geq 90°$, then we drop an altitude from c to projection point c' on ae. Triangle $c'ce$ has an exterior hypotenuse, so it requires no further subdivision. The other triangle, acc', has reduced to the third case, in which both angles alongside the subdivided side are acute.

In the third case, the opposite vertex e projects perpendicularly onto the subdivided side ac. Add edges from e to the subdivision points on either side of its perpendicular projection e', as in Lemma 8. This splits ace into either two or three triangles, depending on whether there exist subdivisions on each side of e'. An obtuse triangle with hypotenuse ce can be handled without further subdivision, even if—as will be the case in lower levels of the recursion—side ce is not exterior but only outbound. A nonobtuse triangle without subdivisions—such as a triangle in the middle—can be handled as is by the downwards pass, as Lemma 8 will apply. The last triangle has one fewer subdivision point, so we partition it recursively. (The recursive call starts by dropping an altitude to ae.)

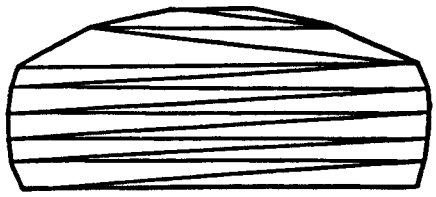

Figure 7. Lower bound example for refinement.

Notice that the upward pass only leaves behind subdivision points that do not propagate beyond their original triangle in the downward pass. In other words, each subdivision on ac either drops an altitude to ae or is left in a triangle that the downward pass processes towards exterior side ce. Thus the downward pass results in $O(n^2)$ complexity overall. ∎

6. Conclusions and Open Problems

We have shown how to triangulate arbitrary polygons using a polynomial number of right and acute triangles. This result demonstrates a strong separation between the complexities of two desirable properties of finite element meshes. Forbidding angles larger than 90° incurs only polynomial cost in size, but forbidding angles smaller than a fixed bound incurs cost dependent upon the geometry of the domain. There are still a number of open problems in "mesh generation theory".

First of all, we would like to extend the results of Section 5 to refining arbitrary straight-line planar graphs. In particular, we do not know how to handle polygons with line-segment holes so that Steiner points on each side of a line-segment hole match up. As mentioned above, another special case of refining planar graphs is simultaneously triangulating the inside and outside of a polygon.

Open Problem 1. *Can every straight-line planar graph be refined into a polynomial-size nonobtuse triangulation?*

Second, the question of lower bounds is interesting. We believe the correct bound to be quadratic.

Open Problem 2. *Can any nontrivial lower bound be shown for the size of a nonobtuse triangulation?*

Paterson observed [12] that for the problem of refining a given triangulation, the complexity must be quadratic, even for convex polygons and a largest angle bound arbitrarily close to 180°. His example is a convex polygon with an "accordion" triangulation, shown in

Figure 7. There are $\Theta(n)$ long, skinny, nearly horizontal triangles; the polygon also has $\Theta(n)$ vertices evenly spaced along the top of this stack of triangles. Because each vertex at the top must have a "downwards" edge, there must be a path downward from each vertex through the stack of triangles. Separate paths cannot merge because of the angle bound. Thus each path has complexity $\Omega(n)$, and the total triangulation has complexity $\Omega(n^2)$. Salzberg et al. gave a similar lower-bound example for their learning problem [13].

Third, it would be interesting to give an algorithm that uses only acute angles, so that all circumcenters lie in the interiors of their elements. By carefully warping rectangles and right triangles, it is possible to turn our linear-size nonobtuse triangulations of point sets into acute triangulations (included in the journal version of [2]). For polygon input, however, some new ideas are needed, as it is not possible to divide an obtuse triangle into acute triangles, adding subdivision points only to the base.

Finally, there remains the question of extending these results to higher dimensions. We believe the correct analog of a nonobtuse triangle is a simplex that contains its circumcenter; this generalization most closely follows the mesh generation application. Perhaps the size bound for a nonobtuse triangulation of a d-dimensional polyhedron will turn out to be $O(n^d)$, matching a lower-bound example for the inside-outside triangulation problem [13].

Acknowledgements

We would like to thank Warren Smith for drawing our attention to this problem, and David Dobkin, John Gilbert, and Mike Paterson for some helpful discussions.

References

[1] B. S. Baker, E. Grosse, and C. S. Rafferty. Nonobtuse triangulation of polygons. *Discrete and Comp. Geom.*, 3:147–168, 1988.

[2] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. In *31st Symp. Found. Comp. Sci.*, pp. 231–241. IEEE, 1990.

[3] M. Bern and J. Gilbert. Drawing the planar dual. Manuscript, 1991.

[4] L. P. Chew. Constrained Delaunay triangulations. In *3rd Symp. Comp. Geom.*, pp. 215–222. ACM, 1987.

[5] H. Edelsbrunner, T. S. Tan, and R. Waupotitsch. A polynomial time algorithm for the minmax angle triangulation. In *6th Symp. Comp. Geom.*, pp. 44–52. ACM, 1990.

[6] D. Eppstein. The farthest point Delaunay triangulation minimizes angles. Manuscript, 1990.

[7] I. Fried. Condition of finite element matrices generated from nonuniform meshes. *AIAA J.*, 10:219–221, 1972.

[8] J. L. Gerver. The dissection of a polygon into nearly equilateral triangles. *Geom. Dedicata*, 16:93–106, 1984.

[9] D. T. Lee and A. K. Lin. Generalized Delaunay triangulation for planar graphs. *Discrete and Comp. Geom.*, 1:201–217, 1986.

[10] D. T. Lee and B. J. Schachter. Two algorithms for constructing a Delaunay triangulation. *Int. J. of Computer and Information Sciences*, 9:219–242, 1980.

[11] D. Mount and A. Saalfeld. Globally-equiangular triangulations of co-circular points in $O(n \log n)$ time. In *4th Symp. Comp. Geom.*, pp. 143–152. ACM, 1988.

[12] M. S. Paterson. Personal communication, 1990.

[13] S. Salzberg, A. Delcher, D. Heath, and S. Kasif. Learning with a helpful teacher. To appear in *12th Int. Joint Conf. on Art. Intelligence*, Sydney, Australia, 1991.

[14] R. Sibson. Locally equiangular triangulations. *Computer J.*, 21:243–245, 1978.