Subquadratic Nonobtuse Triangulation of Convex Polygons

David Eppstein

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

Tech. Report 91-61

July 15, 1991

Abstract

A convex polygon with n sides can be triangulated by $O(n^{1.85})$ triangles, without any obtuse angles. The construction uses a novel form of geometric divide and conquer.

1 Introduction

A number of recent papers [1, 2, 3, 4, 6, 8] have studied *Steiner triangulation* problems in which the angles in the triangulation are limited in some way.

A Steiner triangulation of a planar straight line graph (PSLG) is a collection of triangles that cover the graph and meet face to face. Edges of the graph may be subdivided in the triangulation, but they must be covered by triangle edges. Vertices of the graph must be covered by triangle corners; the remaining triangle corners are called *Steiner points.* The goal is typically to find a triangulation minimizing the number of triangles, or equivalently minimizing the number of Steiner points.

Without further restriction, any PSLG can be triangulated without introducing any Steiner points. However the problem becomes considerably more difficult if the angles used in the triangles must lie in some range $\alpha \leq \theta \leq \beta$. If $\alpha > 0°$, or if $\beta < 90°$, then even for such simple inputs as point sets and convex polygons, the number of triangles must depend not only on the size of the PSLG, but also on its geometry [4]. For this reason no polynomial time triangulation algorithm is possible in this case. However algorithms are known for these problems [1, 4, 6], which run in time polynomial in the combined input and output size, and which in some cases use a number of triangles within a constant factor of the minimum possible [4].

Here we consider the problem that arises when the angles must not be obtuse; in the notation of the previous paragraph, $\alpha = 0°$ and $\beta = 90°$. This problem has applications in finite element mesh generation, both due to the numerical properties of finite element systems based on nonobtuse triangles [1] and due to the simple relation of such a triangulation with its dual [1, 3, 5]. There is also a more geometric motivation for this problem: a non-obtuse triangulation must be the Delaunay triangulation of its vertices; therefore, if we triangulate the inside and outside of a polygon with nonobtuse triangles, we cover the edges of the polygon with Delaunay edges. It is also the case that, by placing sites near the corners of such a triangulation, the polygon edges can be covered by boundaries of cells in a Voronoi diagram [8]; such a cover has applications in computational learning theory [8].

Several types of PSLG are known to have polynomial size nonobtuse triangulations. A point set can trivially be triangulated with $O(n^2)$ right triangles. Polygons can be triangulated with $O(n^2)$ non-obtuse triangles [3]. If the PSLG is a triangulation of a simple polygon, $O(n^4)$ triangles suffice [3]. For certain other PSLGs, we can prove $\Omega(n^2)$ lower bounds. This includes nonobtuse triangulation of PSLG's formed by triangulating (con-

vex) polygons [3, 7], simultaneous triangulation of the inside and outside of a (non-convex) polygon [8], and triangulation of a collection of disjoint edges. In fact these lower bounds hold for any $\beta < 180°$. Therefore (except for the $O(n^4)$ bound, which seems unlikely to be tight) it seemed that the correct complexity for many of these problems was $\Theta(n^2)$.

Doubt was cast on this by an earlier result of Bern et al. [4]. They showed that an algorithm based on quadtrees could triangulate a point set using only $O(n)$ acute triangles. In this paper we give the second such subquadratic nonobtuse triangulation result. We show that a convex polygon with n faces can be triangulated using $O(n^{1.85})$ nonobtuse triangles. Our algorithm starts by forming an $O(n^2)$-size triangulation based on a method of Bern and Eppstein [3], selects portions of the triangulation resembling rectangular grids, and then "thins" those grids using a novel form of divide and conquer.

Other nonobtuse triangulations of Bern and Eppstein [3] contain similar rectangular and fan-shaped grids of triangles. Therefore it is likely that the methods of this paper can be extended to improve the bounds for triangulating arbitrary polygons and other PSLG's.

2 Initial quadratic triangulation

We first describe the initial stages of our algorithm, in which we form a grid of $O(n^2)$ rectangles, together with $O(n)$ triangles along the border of the grid. This is essentially the same as a method described by Bern and Eppstein [3], but we repeat the description for completeness.

We start by partitioning the polygon vertically and horizontally. We later refine the grid by adding more horizontal and vertical lines, in order to be able to triangulate the region between the grid and the polygon boundary.

The choice of horizontal and vertical axes is made so that the longest diagonal of the polygon (called the *main diagonal*) is oriented horizontally. 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 our construction can be somewhat simplified.

First, draw a vertical line segment through each vertex of the polygon, extending to the boundaries of the polygon. These lines divide the polygons into quadrilaterals with two vertical sides. Each vertex of a quadrilateral will either be an original input vertex, or a point where a vertical line touches the polygon boundary. Draw a horizontal line segment through each such point; extend the line segment to the last possible vertical segment. In other words,

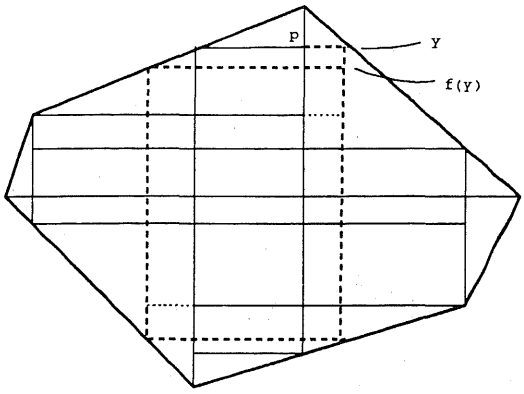

Figure 1. Convex polygon with path from subdivision point.

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.

Our algorithm 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 the following fact.

Lemma 1. *If a right triangle has one subdivision point on a leg, it can be triangulated with three right triangles, by adding a new subdivision point on the hypotenuse.* □

If a subdivision point is alone in its triangle, then this will clearly remain the case for the corresponding subdivision point after an extension of the partition. Thus our strategy will be simply to 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 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 1 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 2. *The function $f(y)$ is continuous, monotone, and piecewise linear with $O(n)$ breakpoints in linearity.*

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 3. *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. We describe the current state of the triangulation.

Lemma 4. *Given a convex polygon, we can partition it into $O(n^2)$ rectangles and $O(n)$ triangles meeting face to face. The rectangles can be divided into four subsets, each of which is bounded by a horizontal line, a vertical line, and a monotonic "staircase" which makes steps of one rectangle horizontally, and either one or two rectangles vertically.*

Proof: The rectangles can be divided horizontally along the long diagonal; each subset then can be divided into two pieces at the point of maximum vertical extent. This gives the desired partition into four subsets. The $O(n)$ right triangles around the boundary each have at most a single subdivision on their vertical sides, from which fact the description of the "staircase" boundary then follows. Each boundary triangle can be further divided into three unsubdivided right triangles by Lemma 1. □

3 Finding rectangular subproblems

Our triangulation now proceeds as follows. We form a grid as above; from this grid we pick out larger rectangles. Each large rectangle will be divided into smaller rectangles, say $a \times b$ of them. Then the large rectangle will have $a-1$ subdivisions each on its left and right boundaries, and $b-1$ subdivisions each on the top and bottom boundaries, for a total of $2(a+b-2)$. The subdivisions on opposite sides of the large rectangle match up to each other. Our strategy will be to triangulate each large rectangle separately. We are allowed to do anything we want in the interior of the large rectangle, but the boundary must end up with exactly the subdivisions specified. Obviously we could triangulate the regions with $O(ab)$ triangles; we improve on that in the next section. Here we describe how to find the large rectangles.

Recall that our small rectangles are grouped into regions with one vertical side, say x small rectangles high, one horizontal side, y small rectangles long, and one "stairstep" side. Initially x and y are both $O(n)$. Figure 2 depicts such a region. The small rectangles are drawn as squares. The stairstep side here makes steps of one rectangle vertically, and either one or two horizontally; the roles of horizontal and vertical are exchanged from the previous section, so that the figure will fit better on the page.

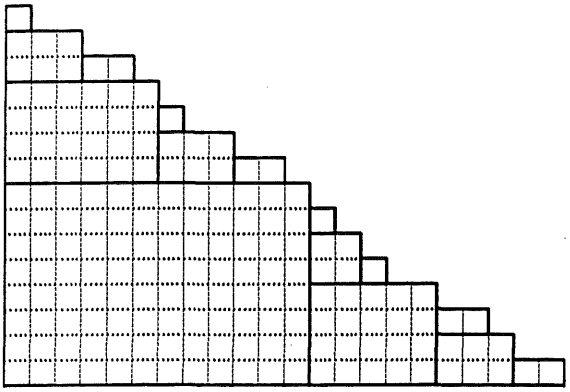

Figure 2. Grouping small rectangles into larger rectangular regions.

Given such a stair-shaped region of small rectangles, we can find one large rectangle easily as follows. Extend a horizontal line at the level corresponding to the $x/2$ row of rectangles. Drop a vertical where this horizontal hits the stairstep boundary. This leads to a large rectangle that is $x/2$ small rectangles high, and between $x/2$ and x rectangles long. The remainder of the small rectangles are in two stairstep regions, above and to the right of the large rectangle, each of height $x/2$. If we repeat this process recursively, we get one large rectangle of height $x/2$, two of height $x/4$, and in general 2^i rectangles of height $x/2^{i+1}$. Thus we have the following lemma.

Lemma 5. *If a rectangle with m subdivisions on its sides, and with matching subdivisions on opposite sides, can be triangulated using $R(m)$ nonobtuse triangles, without introducing any new subdivisions, then any convex polygon can be triangulated with $O(T(m))$ nonobtuse triangles, where $T(m)$ satisfies the recurrence*

$$T(m) = R(m) + 2T(m/2) \tag{1}$$

Proof: We produce four groups of small rectangles, with sides horizontal, vertical, and stairstep, as in Lemma 4. There will be $O(n)$ additional triangles, but this number will be dominated by the solution to Recurrence 1. Then the construction above partitions each group into rectangular regions with heights (measured in numbers of small rectangles) matching the recurrence above, and lengths (again measured in small rectangles) proportional to the heights. Since the number of subdivisions on the boundary of each

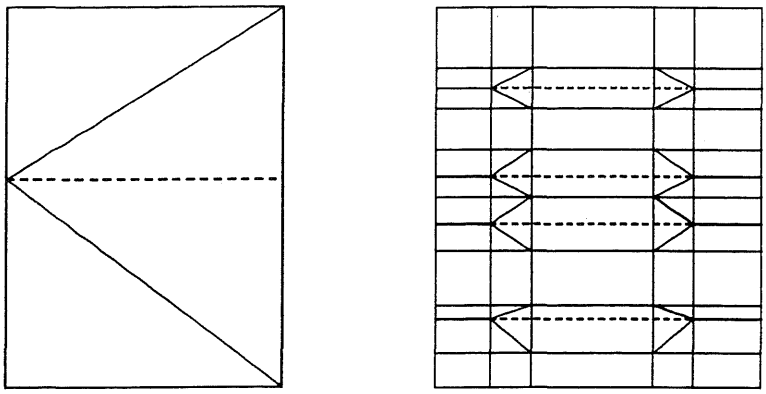

Figure 3. Removing interior lines from subdivided rectangle: (a) Two small rectangles merge, removing a subdivision; (b) previous figure and its reverse combine to remove lines from the grid.

rectangular region is proportional to the height plus the length, the recurrence correctly describes the total time. □

4 Rectangles with subdivisions

Our algorithm for triangulating the large rectangles found in the previous section is based on the following observation.

Lemma 6. *If a rectangle has a single subdivision point on a vertical side, and its width is more than either of the distances between the subdivision and the top or bottom of the rectangle, then the rectangle can be non-obtusely triangulated without introducing more subdivisions.* □

Such a rectangle is illustrated in Figure 3(a). We use this observation to remove line segments from the rectangular grid formed by the subdivisions of our large rectangle.

Let a *column* be a set of vertically adjacent small rectangles; then the width of the column is the distance between the subdivisions that form it. Similarly let a *row* be a set of horizontally adjacent small rectangles, and let the height of a row be the distance between subdivisions that form it.

Then if two columns have widths a and b, and two adjacent rows have heights c and d, and if a and b are both greater than c and d, we can remove

the portion of horizontal line, separating the two rows, between the two columns. This process produces triangles, at the endpoints of the removed line segment, that may not be changed by later modifications to the grid. In order to gain any benefit from this process, we must remove many horizontal lines using the same columns. Figure 3(b) illustrates several horizontal lines being removed by the same two columns.

We now quantify how many lines can be removed at a time.

Lemma 7. *Let a rectangle be given with m subdivisions on the vertical sides (so it has $m+1$ rows). Let a and b be the widths of two columns, with $\min(a,b)$ at least as large as $(1/2+\epsilon)m$ of the row heights. Then $\epsilon m - 1$ horizontal lines can be eliminated between the columns.*

Proof: There are $(1/2-\epsilon)m+1$ rows that are too high to satisfy Lemma 6, and each one protects the lines above and below it. Thus there are $m-2\epsilon m+2$ lines protected, so $2\epsilon m - 2$ lines that can be removed. In the worst case, the removable lines will be all adjacent to each other, and we will be able to remove at most half of them. Therefore we can remove $\epsilon m - 1$ lines. □

We are now ready to describe our algorithm. As before, suppose there are m rows and n columns. Let x be the 2/3-median of the row heights; i.e., $m/3$ rows are higher than x and $2m/3$ are less high. Similarly let y be the 2/3-median of the columns.

Assume $x < y$; the opposite case is treated symmetrically. Let a be the leftmost column with width at least x and let b be the rightmost such column. Then because $n/3$ rows have height at least $y > x$, columns a and b are separated by at least $n/3$ columns. Also note that a and b are wider than the height of $2m/3$ rows; therefore by Lemma 7 we can eliminate $m/6$ rows between the two columns.

This gives us three subproblems, which are solved recursively. The subproblems consist of (1) the columns between 1 and $a-1$, with all m rows; (2) the columns between $a+1$ and $b-1$, with $5m/6$ rows; and (3) the columns between $b+1$ and n, with all m rows. We also triangulate columns a and b, using m triangles. This partition and row elimination is shown in figure 4(a). The total number of triangles used is

$$R(m,n) = R(a-1,m) + R(b-a-1,5m/6) + R(n-b,m) + m.$$

If we solve for the worst case time, by allowing all possible choices of a and b (with $b-a \geq m/3$) and by including the dual case in which $x > y$,

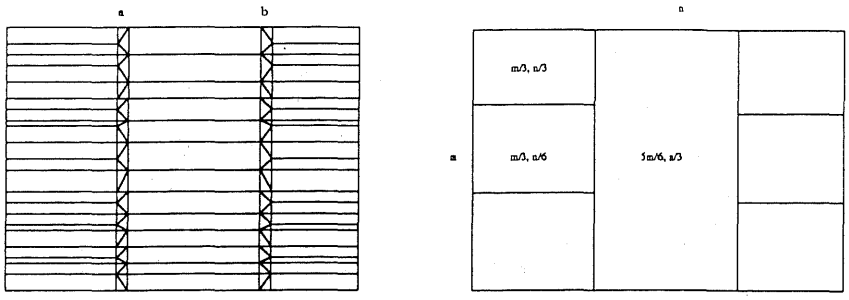

Figure 4. Recursive division into smaller rectangles: (a) single split, split into three parts with fewer rows in middle; (b) outer subproblems split again, eliminating columns.

we achieve a bound that is slightly better than linear. Instead of solving this recurrence, we first make an observation that allows us to improve our algorithm and hence our bounds.

Let us examine more carefully subproblems (1) and (3). By construction, all columns in the subproblems have widths smaller than x, which is still the 2/3-median of the row heights. Therefore, by choosing the topmost and bottommost rows with heights at least x, we can apply Lemma 7 to eliminate half of the columns between the two rows.

Unfortunately this second level of improved row and column elimination can not be continued to a third level and so on; this failure occurs because the first level split the problem according to *columns* which were chosen larger than the 2/3-median of the *rows*; in the second level it is *rows* that are chosen, but they are compared to the *row* median. So the second level recursion is not the same as the first level, and the trick can not be extended.

Thus we can subdivide each of the two subproblems into three sub-subproblems, giving a recursion which divides each rectangle into seven parts, as depicted in figure 4(b). The recurrence describing the possible arrangements of the seven parts is quite complicated. But as indicated by the numbers in the figure, we can make some simplifying assumptions about the dimensions of each subproblem. We do not attempt to prove these assumptions; instead we use them heuristically to simplify the recurrence, giving a solution as a function of m and n. Then it can be verified that this is also the solution of the true recurrence.

Our assumptions are as follows:

- In the worst case, subproblems (1) and (3) have equal numbers of columns, i.e. $a = n - b$. Similarly, the corresponding sub-subproblems have equal numbers of rows. This follows from the fact that the worst case number of triangles is at most linear in each of n and m.

- In the worst case, subproblem (2) will use exactly $m/3$ columns, and the corresponding sub-subproblems will use exactly $n/3$ rows. This is clear enough for the sub-subproblems, because eliminating extra columns is unlikely to increase the number of triangles. In subproblem (2), again we would expect eliminating rows to decrease the number of triangles. But this would also shrink the area involved in the more efficient second-level recursion. So we must verify that row elimination is the stronger effect.

- If $m < n$ the worst case happens when $x > y$ and the first level recursion splits the problem along rows of the rectangle. If $m > n$ the worst case happens when $x < y$ and the rectangle splits by columns This is heuristically justified as follows. Suppose the problem is split by columns (as depicted in the figures). Then by the previous two assumptions the subproblem sizes are $(m/3, n/3)$, $(m/3, n/6)$, and $(5m/6, n/3)$. In each case the ratio n/m increases in the subproblems compared to the original problem. If we split by rows, the subproblem sizes would give the same values of the products nm, but the ratio n/m would decrease. For a fixed value of mn, we expect more triangles when the ratio n/m is farthest away from 1 (e.g. when $m = 1$ or $n = 1$ the number of triangles must be $\Theta(mn)$, but when $m = n$ we have subquadratic bounds). So the worst case should be the one in which these ratios are farthest from 1, and that will happen when the problems are split as described.

With these assumptions, we can describe the number of triangles used by the following recurrence, for the case $m < n$.

$$R(m,n) = R(m/3, 5n/6) + 2R(m/6, n/3) + 4R(m/3, n/3) + O(n).$$

The first term represents subproblem (2), the second term represents the two corresponding sub-subproblems, and the third term represents the four remaining sub-subproblems.

The recurrence solves to $O(m^{0.8475}n)$. The exponent in the solution was derived numerically. It is an approximate solution to the equation

$$(5/6 + 4 \cdot 1/3)(1/3)^x + (2 \cdot 1/3)(1/6)^x = 1.$$

We show below that this time bound is the true solution to the unsimplified recurrence that actually describes our algorithm. Let us state the results of this section.

Lemma 8. *A rectangle subdivided into m rows and n columns, with $m < n$, can be triangulated using $O(m^{0.8475}n)$ nonobtuse triangles, without introducing any new subdivisions.*

Theorem 1. *Any convex polygon can be nonobtusely triangulated with $O(n^{1.8475})$ triangles.*

Proof: By Lemmas 5 and 8, the number of triangles satisfies the recurrence

$$T(n) = R(n,n) + 2T(n/2) = O(n^{1.8475}) + 2T(n/2).$$

But this solves to $O(n^{1.8475})$. □

5 Complete solution of recurrence

For completeness, we prove that the true recurrence describing our rectangle triangulation algorithm has the solution we claimed. We assume throughout that $m < n$. By induction we can assume $T(m',n') = c(m')^{0.8475}n'$ for some c and for all $m'n' < mn$. This is clearly true for $m = 1$ since then the result is simply $O(n)$, and in fact $2n$ triangles suffice.

First suppose that the initial subdivision is by rows. Then the following recurrence describes the number of triangles needed. Here $m_1 + m_2 + m_3 = m$, $n_1 + n_2 + n_3 = n_4 + n_5 + n_6 = n$, $m_2 \geq m/3$, $n_2 \geq n/3$, and $n_5 \geq n/3$. Note that $m' < n'$ for all subproblems involved, so the inductive hypothesis can be used directly.

$$\begin{aligned} R(m,n) &= R(m_1,n_1) + R(m_1/2,n_2) + R(m_1,n_3) \\ &\quad + R(m_2,5n/6) \\ &\quad + R(m_3,n_4) + R(m_3/2,n_5) + R(m_3,n_6) \\ &\quad + O(n) \\ &= cm_1^{0.8475}n_1 + c(m_1/2)^{0.8475}n_2 + cm_1^{0.8475}n_3 \\ &\quad + cm_2^{0.8475}(5n/6) \\ &\quad + cm_3^{0.8475}n_4 + c(m_3/2)^{0.8475}n_5 + cm_3^{0.8475}n_6 \\ &\quad + O(n) \end{aligned}$$

$$
\begin{aligned}
&= cm_1^{0.8475}n_1 + c(m_1/2)^{0.8475}n_2 + cm_1^{0.8475}n_3 \\
&\quad +cm_2^{0.8475}(5n/6) \\
&\quad +cm_3^{0.8475}n_4 + c(m_3/2)^{0.8475}n_5 + cm_3^{0.8475}n_6 \\
&\quad +O(n) \\
&\leq cm_1^{0.8475}(0.85192n) + cm_2^{0.8475}(0.83334n) \\
&\quad +cm_3^{0.8475}(0.85192n) + O(n) \\
&= c(m_1^{0.8475} + m_2^{0.8475} + m_3^{0.8475})(0.85192n) \\
&\quad -cm_2^{0.8475}(0.01858n) + O(n) \\
&\leq c(1.1824m^{0.8475})(0.85192n) \\
&\quad -c(0.39414m^{0.8475})(0.01858n) + O(n) \\
&\leq 0.99999cm^{0.8475}n + O(n).
\end{aligned}
$$

So if c is large enough the result is $O(m^{0.8475}n)$ as claimed.

In the other case, the initial subdivision is by columns. For any possible such subdivision, we can construct an alternate subdivision by rows, in which the ratios of n_i/n and m_i/m switch roles to match the recurrence above. So in the alternate problems, the products $m_i n_i$ are the same as they would be in the original problems. But the *aspect ratio* n_i/m_i is larger in the alternate subproblems than in the originals. In some cases the original aspect ratio may even be smaller than one (i.e., in the original problem, n_i may be smaller than m_i). But in the latter case m_i/n_i will be smaller than the alternate value of n_i/m_i. In the inductive hypothesis, reducing the aspect ratio and fixing the product $m'n'$ can only decrease the total number of triangles. So subdivision by columns is preferable to subdivision by rows, and cannot occur as the worst case.

6 Conclusions

We have shown that $o(n^2)$ nonobtuse triangles suffice to triangulate any convex polygon. Our algorithm can be somewhat improved; in particular, choosing .678-medians instead of 2/3-medians leads to an exponent of 1.84736. Perhaps we should use different medians for the rows and columns, depending somehow on the ratio n/m.

Our method depends only on finding subproblems in the form of a rectangular mesh. If we represent meshes implicitly in terms of their row heights

and column widths, the computation time will be governed by the same recurrence as the number of triangles; hence this too will be $o(n^2)$.

The best known nonobtuse triangulation algorithm for non-convex polygons [4] uses $O(n^2)$ triangles, and forms configurations resembling such meshes; perhaps our algorithm can be extended to work in this case. Recently Bern et al. [2] have shown how to triangulate any polygon with $O(n \log n)$ triangles, with no angle larger than 135°; their algorithm is again based on finding rectangular subproblems, but in their case linearly many triangles suffice for each subproblem. Perhaps their techniques and ours can be combined to yield better combined bounds on angles and numbers of triangles. It at least seems likely that our algorithm can be used to *approximate* a non-obtuse triangulation, by which we mean all angles must be at most $90° + \epsilon$ for some ϵ specified as part of the input.

It is also possible that a different algorithm can solve the same problems more efficiently. Quadtree-based algorithms can give linear sized nonobtuse triangulations of point sets [4]; perhaps quadtrees can be useful for the convex polygon problem as well.

There is another question that arises naturally from this work. The rectangular subproblems we solve can themselves be thought of as convex polygons which we triangulate. However we operate under the further restriction that all Steiner points be interior to the polygon; none are allowed to subdivide the boundary. With this restriction, we can still triangulate the polygons in subquadratic complexity. Can we characterize the polygons which can be so triangulated? How many triangles are needed in general?

This question is related to a final open question, raised by Bern and Eppstein [3]. Is it possible to find a polynomial size nonobtuse triangulation of an arbitrary planar straight line graph? One way of achieving this would be to refine the graph until all regions are triangulable without adding more boundary subdivisions; therefore it is important to know what sorts of regions are possible, and how many triangles are needed in each.

References

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

[2] M. Bern, D. Dobkin, and S. Mitchell. Manuscript in preparation.

[3] M. Bern and D. Eppstein. Polynomial size non-obtuse triangulation of polygons. 7th ACM Symp. Comput. Geom. (1991) 342–350.

[4] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. 31st IEEE Symp. Found. Comp. Sci. (1990) 231–241.

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

[6] L.P. Chew. Guaranteed-quality triangular meshes. Report TR-89-983, Cornell U., 1989.

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

[8] S. Salzberg, A. Delcher, D. Heath, and S. Kasif. Learning with a helpful teacher. 12th Int. Joint Conf. Artificial Intelligence (1991) to appear.