Carnegie Mellon University
Research Showcase @ CMU

Computer Science Department | School of Computer Science

1-2004

A Time Efficient Delaunay Refinement Algorithm

Gary L. Miller
Carnegie Mellon University, glmiller@cs.cmu.edu

Follow this and additional works at: http://repository.cmu.edu/compsci

Published In
.

This Conference Proceeding is brought to you for free and open access by the School of Computer Science at Research Showcase @ CMU. It has been accepted for inclusion in Computer Science Department by an authorized administrator of Research Showcase @ CMU. For more information, please contact research-showcase@andrew.cmu.edu.

A Time Efficient Delaunay Refinement Algorithm *

Gary L. Miller†

Abstract

In this paper we present a Delaunay refinement algorithm for generating good aspect ratio and optimal size triangulations. This is the first algorithm known to have sub-quadratic running time. The algorithm is based on the extremely popular Delaunay refinement algorithm of Ruppert. We know of no prior refinement algorithm with an analyzed sub-quadratic time bound. For many natural classes of meshing problems, our time bounds are comparable to know bounds for quadtree methods.

1 Introduction

Generating good aspect ratio meshes from input constraints is possibly one of the most important applications in computational geometry. The goal is to find a triangulation of the input domain satisfying the following three conditions: 1) **Conforming:** The triangulation should conform to the input boundary constraints. 2) **Good Aspect Ratio:** No triangle should contain a small angle. 3) **Size Optimal:** It should contain a minimum number of triangles. There are several different approaches to the fundamental problems that are used both in theory and in practice. The approaches include advancing front, quad-tree, and Delaunay refinement [BE92]. At the present time the smallest meshes come from Delaunay refinement especially for complicated input boundaries. On the other hand, quadtree methods are the only methods known to have sub-quadratic algorithms and optimal size up to constants. In this paper we shall show that we can have the best of both: There are Delaunay refinement algorithms that are asymptotically faster than known quadtree algorithms but with smaller size meshes for a very natural class of meshing problems.

Having a timing analysis for a Delaunay refinement algorithm is important. One, at the present time Delaunay refinement gives substantially smaller meshes for the same input and thus is the preferred method. Two, Delaunay refinement is widely used in practice and has been reimplemented may times. The Delaunay refinement package "Triangle" has thousands of users [She95] from Graphics, the Sciences, and Engineering. Triangle is a careful implementation of Delaunay refinement and is known to work very well in practice, [She96, She02a]. The worst cases for Triangle, as well as several other implementations, are not known but believed to be worst than that of, say, quadtree methods.

Delaunay refinement has been proposed by several researchers. Two important works are those of Chew and Ruppert [Che89, Rup95]. Ruppert was the first to show size optimality [Rup95] but he required that all input angles were at least 90 degrees. More recent papers have addressed the small input angle condition as well as other improvements [She97, She02b, She00, BOG01, Kad01]. None of these improvements, however, included a sub-quadratic running time analysis.

Ruppert [Rup95] proposed a general scheme for refining a Delaunay triangulation of a 2D mesh. His 2D Delaunay refinement algorithm handles inputs that include edges, namely, Planar Straight Line Graphs (PSLG). His paper, as well as subsequent papers, do not fully specify how important aspects of the refinement algorithm should be implemented to get an efficient algorithm. These aspects can dramatically effect its efficiency. Naive implementations of the algorithm, as presented by Ruppert, have quadratic worst-case running time. One important exception is the parallel algorithm of Spielman, Teng, and Üngör who give a parallel incremental algorithm that runs in $O(\log^2(L/S))$ parallel time, where L and S are the largest and smallest input features [STÜ02]. But they do not give sequential time or work bounds for their algorithm. Ruppert's algorithm has quadratic worst-case running time even when L/S is bounded by a polynomial in the input size [Bar02]. The number L/S is often referred to a grading of the input or mesh.

Given the dearth of timing analyses for Delaunay refinement, some have proposed we just implement quadtree-based methods since they have timing analyses and optimal size meshes[BEG94, BET99]. Bern, Eppstein, and Teng give a quadtree algorithm with a running time of $O((n \log n + m) \log m)$ where n is the input size and m is the output size[BET99]. But in prac-

*A complete version can be found at http://www.cs.cmu.edu/ glmiller

†Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213 (glmiller@cs.cmu.edu). Supported in part by NSF Grants CCR–9902091, CCR-9706572, and ACI-0086093.

tice quadtree meshes are much larger for the same input than Delaunay refinement meshes. We will show that one can get a Delaunay quality mesh in the same asymptotic time bounds as those bounds known for quadtree methods. Notice that the timing bounds for quadtree and Delaunay are incomparable since one is in output size and other is in grading. In order to compare these different algorithm types we will need to set bounds on these two terms.

The idea behind Delaunay refinement is very simple: Starting from a initial Delaunay triangulation, whenever we have a triangle with a "bad" aspect ratio, we either add the center of the circle that goes through the vertices of the triangle, the triangle's circum-center, or add the center of a nearby boundary segment. The main issue we try to address in this paper is how to take this algorithm "specification" and make it into an efficient algorithm with a provable timing analysis. As a simple example, what order should the circum-centers be added to the mesh to minimize the work done? Even though all orderings have the same quality guarantees[Rup95], if we add the circum-centers in the wrong order, we may perform dramatically more work than the optimal order. The main reason for the extra work is that the addition of each new circum-center may require us to remove a constant fraction of the old triangles and replace them with new triangles, giving us an algorithm that has quadratic runtime in terms of the size of the output even when the grading of the mesh is polynomial on the input size. This is only one of many algorithm design decisions that we made to get an efficient algorithm.

We have designed our refinement algorithm to change the algorithm only when we could find no other way to get the analysis to work. We list here some of the small but important changes we have made to the refinement algorithm: First, we always add the circum-center from the largest diameter first and we do not necessarily split edges before adding the circum-centers of triangles. We introduce a priority queue that includes both triangles and edges. Second, the intermediate meshes are constrained Delaunay triangulations. Third, we changed when and under what conditions the insertion of the circum-center of a skinny triangle yields to the center of a segment. These changes required only minor modifications our Delaunay refinement code at CMU. [1]

We state a weaker form of our main theorem 2.1 so that we may compare our time bound to those of other algorithms. In the theorem below n is the number of input points, m the number of points in the final triangulation including the input points, and where Γ is in some sense a localized version L/S as used by Spielman, Teng, and Üngör [STÜ02], in particular, it is as most L/S.

THEOREM 1.1. *The procedure DELAUNAY-REFINEMENT (see Table 1) runs in time* $O((n \log \Gamma(G) + m) \log m)$ *where* m *is the total number of points in the output. The function* Γ *is defined in Definition 2.3*

A very conservative estimate for $\Gamma(G)$ is that it is bounded by a polynomial in n, the input size. If we make this assumption then we get the bound $O((n \log n + m) \log m)$ which is identical to the bound for quadtree [BET99]. Theorem 2.1 states a strong form of this theorem. In this case the dominate term in our analysis is for the cost of the priority queue used to find the next skinny triangle or edge to process. If we ignore the priority queue cost our bound becomes $O(n \log n + m)$ which would be optimal.

The outline of the paper is as follows: In Section 2 we introduce both the basic and critical definitions used in both the algorithms and their analysis. This section also includes the main theorem. In Section 3 we give a high-level description of the algorithm, as well as, pseudo-code for the major components. In Section 5 we present a several lemmas that we need to prove that both the algorithm runs in the time claimed and is size optimal. The timing analysis is in Section 4 while the proof that the mesh is optimal size is in Section 6. Finally, in Section 7 we give some concluding remarks as well as open questions.

2 Preliminaries

Throughout this paper, unless explicitly stated otherwise, all objects are in $\mathbb{R}^2$. Given a set S, the **convex closure** of S is the smallest convex set containing S, while the **convex hull** is the boundary of the convex closure of S. In our case the convex hull will always be a cycle of edges in counter-clockwise order, **CCW**. All **disks** are open and the boundary of a disk is called a **circle**. We assume that all circles have a well-defined **interior**. A **simplex** is the convex closure of an affine independent set of points. Given a **simplex** (an edge or a triangle) it has a well-defined **circum-circle** (the minimum radius circle) containing its points. It in turn has a well-defined **circum-radius** and **circum-center**. The **diameter** of a compact set is the maximum distance between any pair of points in the set.

We say that a point p **encroaches** on a simplex S if it is interior to the circum-circle of S. More generally,

[1] The algorithm presented and analyzed here is simpler than earlier versions of this paper and with better time bounds. This algorithm does not include a messy preprocessing step needed for in the prior analysis.

a simplex S **encroaches** on a simplex S' if the circum-center of S encroaches on S'.

A **Planar Straight Line Graph (PSLG)** is a set of n points P and a collection of non-crossing straight-line segments with end points in P, a one-dimensional simplicial complex. A point p is **visible** to the point q, with respect to a PSLG G, if the open edge segment from p to q is disjoint from G or contained in an edge of G. More generally, for a cell complex the open edge must be contained in a cell from the complex. Otherwise, we say that p is **invisible to or occluded from** q. We will assume throughout that no four points are co-circular. This restriction can be remove with the use of Delaunay polygons.

A crucial definition is that of **local feature size** as defined by Ruppert [Rup95]. Let $\text{lfs}_G(p)$ be the minimum distance from p to two disjoint simplices in G.

Most papers in the area give a discussion on how to construct a bounding box for the PSLG input. For simplicity and generality we will assume that the PSLG includes its own "bounding box". That is, a PSLG G is said to be a **PSLG with boundary** if 1) G contains its convex hull[2], 2) no edge in the convex hull is encroached by any point from G, and 3) G contains at least three non-collinear points. Thus, we can view G as a 2D cell complex where the interior of G is the union of 2D cells. We say that the cell complex M' is a **refinement** of the cell complex M if each cell in M can be written as the union of cells from M'. Given any point p in the convex closure of G we consider the lowest dimensional cell containing p. We call this the **containing dimension of** p, denoted $\text{CD}(p)$.

Given a PSLG G and a refinement M of G, an edge of M that is contained in an edge of G is called a **segment** or a **constraining edge**.

Given a set of points P, a triangulation M of P is said to be **Delaunay** if no triangle of M is encroached by any point from P. Given a PSLG G, triangulation M, and triangle T of M, we shall say T is **encroached in the constrained sense** (or simply encroached if there is no confusion) by a point p if 1) p is interior to the circum-circle of T and 2) p is visible to all three vertices of T in G. More generally we say that a circle C is **encroached with respect to a point** p if there is a vertex from M interior to C and visible to p. If no triangle of M is encroached, we say that M is a **constrained Delaunay triangulation**.

We shall need to modify the definition of the radius of a circle for our application. We shall call it the visible-radius of a circle. Let G be PSLG, C a circle, and p some point on C. We further assume that C is not encroached by any vertex visible to p. In this case, the **visible-radius** of C with respect to p is the radius if the center is visible to p otherwise it is half the length of the first cord occluding the center, i.e., its is half the diameter of the region visible to p and interior to C, see Figure 1. In general, p may have more than one visible region each of which is contained in the interior of C. In this case, its visible-radius will be the maximum over these regions. We will only be interested in those regions interior to the convex-hull of G. We make explicit the definition of the visible-radius of a triangle.

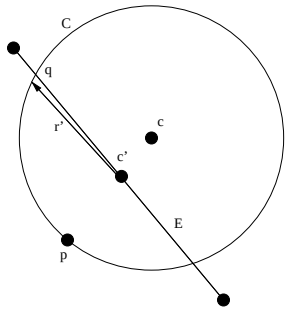

Figure 1: The Visible-Radius of a circle C with respect to a point p.

DEFINITION 2.1. *The* **visible-radius** *of a triangle T in a refinement M of a PSLG G is the minimum visible radii of the vertices of T with respect to its circum-circle denoted by $vr(T)$.*

We can now give a new definition of a spacing function which we will need to give our new upper bound on the time for Delaunay refinement as well as necessary for the analysis. We call this new function the radial-spacing function.

DEFINITION 2.2. *Let M be a refinement of G a PSLG with boundary, and p be a point (not necessarily a vertex of M) in the convex hull of G. The* **radial-spacing** *$R_M(p)$ is the maximum visible-radius of any circle* C *satisfying the following properties:*

1. *p lies on the circle* C*,*
2. C *is not encroached by any point in M with respect to p.*
3. *We only consider visible regions interior to $CH(G)$*

DEFINITION 2.3. *Let M be a refinement of G a PSLG with boundary, and let p be a point (not necessarily a*

[2]The fact that the boundary is convex is most likely not crucial and should be eliminated but we have not worked out the details here.

vertex of M) in the convex hull of G. The **grading** $\Gamma_M(p)$ *is:*

$$\Gamma_M(p) = \frac{R_M(p)}{\text{lfs}_M(p)}.$$

Further, let $\Gamma(M) = \max_{p \in V(M)} \Gamma_M(p)$.

Observe that any mesh is also a PSLG and thus Γ is well defined. We will apply Γ to intermediate meshes to bound the running time.

We can now state the main theorem of the paper.

THEOREM 2.1. *The Delaunay refinement algorithm Table 1 on input a PSLG G with boundary runs in time $O(m + \sum_{p \in V(G)} \log \Gamma_G(p))$ where m is the total number of points in the output.*

The basic approach to Delaunay refinement is to pick a constant angle α_0 and whenever there is a triangle T with angle smaller than α_0 we attempt to add the circum-center of T. This may in turn cause us to add the circum-center (midpoint) of an input segment. But in either case we are adding Steiner points to the mesh. We shall say that a triangle is **skinny** if it contains an angle less than α_0. Otherwise we say the triangle is **fat**.

If M is a Delaunay Triangulation then the **cavity** of a point p (not necessarily in M) is the union of all the triangles of M that are encroached by p. The introduction of the point p into M will form a set of triangles all common to p. We call this configuration a **tent** and the edges common to p the **ridges**. We also refer to the triangles common to a point p as the **star** of p denoted by **Star(p)**.

We also need to pin down a few types of searches for planar triangulations. Suppose that (x, y, z) are the vertices of a triangle T in CCW order. Given the ordered edge xEz we define **Search(E)** to be the ordered pair of ordered edges $[(x, y), (y, z)]$. Given a triangle T and a point p in the circum-circle of T. We define the **path from** T **to** p be the sequence of triangles $T = T_1, \ldots, T_k$ such that 1) either $p \in T_k$ or the edge occluding p from the interior of T_k is a constraining edge and 2) T_{i+1} is the unique triangle sharing an edge e of T_i such that e occludes the interior of T_i from p for $i < k$. Observe that p encroaches each of the triangles on the path.

3 Algorithm

In this section we give our Delaunay refinement algorithm, Table 1. The runtime analysis of this algorithm will be present in Section 4.

As we pointed out in the introduction this algorithm follows very closely when possible to the known implementation of Delaunay refinement as proposed by Rupert. There are several important and subtle changes to make the timing analysis go through.

DELAUNAY-REFINEMENT(G, α_0)
1 % G : PSLG with boundary, α_0: critical angle
2 G' = CONSTRAINED-DELAUNAY(interior(G))
3 Make a priority queue Q (tie: edges have priority).
4 Insert skinny triangles with priority diameters.
5 Insert each encroached edge e with priority $\sqrt{2}|e|$.
6 **while** Q not empty **do**
7 S = EXTRACT-MAX(Q)
8 INSERT-CIRCUM-CTR(G, S, circum-ctr(S))

Table 1: The Delaunay Refinement Algorithm.

The main procedure DELAUNAY-REFINEMENT starts by computing the constrained Delaunay triangulation of the input. There are several known algorithms for computing the constrained Delaunay of a set of non-crossing line segments and points. They all have run time of $O(m \log m)$ where m is the number of points and segments, [Che87, WS87, Sei88]. It would be interesting to know if a simple incremental algorithm such as the one by Kau and Mount could be used without effecting the overall run time [KM92].

In steps 3-5 we setup a priority queue so that edges are processed late enough that all the triangles containing the center of an edge are not too big. We also process them soon enough so that we discover at most once that a circum-center encroaches on the segment as described in step 5 of BFS-CAVITY-TO-90-DEGREES below.

The bulk of the work is in procedure INSERT-CIRCUM-CENTER. The procedure must first determine (in the case when the point p is a circum-center of a triangle) if p encroaches on a segment and do this in such a way that all the work can be charged back to the edge if we decide not to add p. Second, if we decide to add p then the cost will be charged to the edges out of p. It will be important that we add p in some cases even if it encroaches on the edge. Call this strong-encroachment.

DEFINITION 3.1. *Let S be a simplex, E a segment, and E_1 and E_2 the two subsegments obtained by splitting E at its midpoint. We say that S* **strongly-encroaches** *E if 1) S encroaches E and 2) either E encroaches S, S encroaches E_1, or S encroaches E_2.*

We next describe our procedure INSERT-CIRCUM-CENTER for inserting the circum-center of an edge or a triangle into the mesh, see Table 2.

```
INSERT-CIRCUM-CENTER(G, S, p)
 1  % G : PSLG, S : Simplex, p : circum-center(S)
 2  S' ← CONTAINING-OR-OCCLUDING(G, S, p)
 3  if S' is occluding then
 4    QUEUE(Q, S'); EXIT
 5  BFS-CAVITY-TO-90-DEGREES(G, S', S, p)
 6  Boundary ← REMOVE-CAVITY(G, S', p)
 7  FORM-TENT(G, Boundary, p)
 8  QUEUE(Q, skinny triangles in Star(p))
```

Table 2: Procedure repeatedly adds circum-center of skinny triangles and encroached segments.

If S is a segment of G then p is contained in S and procedure CONTAINING-OR-OCCLUDING simply returns S. Otherwise the procedure simply traverses the path from S to the simplex containing p and if it finds a segment e in G which occludes S from p it return this segment instead.

Procedure BFS-CAVITY-TO-90-DEGREES search for any segments in G which p encroaches. It must perform the search in such a way that all the work can be charged to the encroached edge in the case when the simplex S must **yield** to the strongly-encroached edge. One critical point and reason for the term "90-DEGREES" is that if p encroaches on an edge at step 5 the the angle formed at p by the endpoints of E and p must be at least 90 degrees. Also recall, that the convex-hull of a simplex is its boundary in counterclockwise order. The procedure BFS-CAVITY-TO-90-DEGREES is given in Table 3.

```
BFS-CAVITY-TO-90-DEGREES(G, S, S', p)
 1  % G : PSLG, S, S' : Simplex, p : circum-center(S)
 2  Init a FIFO queue with CONVEX-HULL(S')
 3  while FIFO ≠ ∅ do
 4     xEy ← POP(FIFO)
 5     if ENCROACH(p, E) then
 6       if E is segment of G then
 7         QUEUE(Q, E)
 8         if 2 = dim(S) and
 9            STRONGLY-ENCROACH(S, E) then
10           EXIT(INSERT-CIRCUM-CENTER)
11           PUSH(FIFO, SEARCH(E))
```

Table 3: Procedure searches the cavity looking for an encroached edge.

4 Timing Analysis

In this section we analyze the time for procedure DELAUNAY-REFINEMENT.

The algorithm consists of two major stages: 1) computing a constrained Delaunay triangulation, step 2, and 2) the refinement step, the while loop in line 6. The time for constrained Delaunay is $O(n \log n)$ as previously mentioned.

The refinement has three major costs 1) determining strong-encroachments (yield), 2) actually work to insert the points, and 3) maintaining the priority queue. We first consider the cost to yield.

4.1 Paying for the Yields If we insert a point p into the mesh we shall charge the ridges out of p for all the work performed to insert p. On the other hand, in the case when the circum-center of a triangle is not added but instead yields to a strongly-encroached segment, we will charge all the work to discover the strong-encroached segment to the segment.

LEMMA 4.1. *If T is a triangle with circum-center p then all the triangles on the path from T to p belong to the chain containing T. Each triangle is encroached by p. If T is a maximum diameter skinny triangle then they are all fat triangles.*

Proof. The triangles on the path except for possibly the last one are obtuse and thus strictly increasing in diameter. The fact that each triangle is encroached by p follows by induction on the number of triangles in the chain. ∎

Since all the triangles on the path are in the cavity of p and thus by Lemma 5.2 there is at most a constant number of triangle on the path. Therefore procedure CONTAINING-OR-OCCLUDING takes at most a constant amount of work per call.

We next analyzes the cost for procedure BFS-CAVITY-TO-90-DEGREES(G, S, S', p) in the case when p is not added to the mesh.

LEMMA 4.2. *If T is a triangle and E is a segment such that E encroaches T, then any triangle on the path from T to the midpoint of E is encroached by E.*

Proof. The proof is by induction on the number of triangles on the path. ∎

Thus, if a point yields due to condition 1) of the definition of strong-encroachment, the cost to search all the triangles on the path can be charged to the segment since when adding the segment's circum-center all these triangles will be removed. The searching is done in a BFS manor with a queue of size at most four. Thus, we can charge all the triangles search to the segment as well. This charge will be made at most once when the segment is added to the queue. We claim that at most one triangle will be found to encroach on an edge E since a triangle that encroaches on E must have priority less

than that of E. But, we process edges and triangles highest priority first.

We next consider condition 2) of the definition of strong-encroachment. Here we will show that there is at most a constant number of triangles searched and thus we charge this work to the segment. It will suffice to prove the following lemma:

LEMMA 4.3. *If T is a triangle, T' is the triangle containing the circum-center of T, and E a segment such that T encroaches E, T encroaches on a subsegment E_1 of E, and E does not encroach T then any triangle on the path from T' to the midpoint of E has diameter* $>$ dia(T)

Proof. Let C and p be the circum-circle and circum-center of T. Let C', p', and r' be the circum-circle, circum-center, and circum-radius of E respectively. Any triangle from T' to E must have an edge intersecting the line $p - p'$. This edge cannot have an endpoint in the interior of either C or C'. We claim that the length of such an edge is $> 2r$.

The proof of the claim follows by observing that line segment between the points $C \cap C'$ does not occlude p from p' and that $r < r'$. It follows that the shortest such edge is a chord of C' with midpoint p. Therefore it must be longer than $2r$. ∎

4.2 The cost of Insertion Using Subsection 4.1 the cost of all the calls to INSERT-CIRCUM-CENTER, step 8 of DELAUNAY-REFINEMENT, can be upper bounded by the number of ridges in the directed graph $\mathcal{G}$ as defined in Subsection 5.1. We bound the number of ridges by bounding the indegree of $\mathcal{G}$. We state this as a theorem:

THEOREM 4.1. *Let p be a point either in the input graph G or added to G. Suppose that M is G if $p \in V(G)$ otherwise let M be the triangulation just after adding p. Then the indegree of p in $\mathcal{G}$ is at most $C_{4.1} \cdot \log \Gamma_M(p)$ for some constant $C_{4.1}$*

Proof. Let q be a point with a ridge to p. By Lemma 5.8 we know that the distance from p to q is between $C_{5.8} \operatorname{lfs}_M(p)$ and $2R_M(p)$. Let $A(p, \rho, 2\rho)$ be an annulus centered at p of radius from ρ to 2ρ. Therefore if we show that there can be at most a constant number of points q in $A(p, \rho, 2\rho)$, we will be done.

We count the number points midpoints of segment separately from circum-centers of triangles in the annulus. The case for midpoints of segments was done in Lemma 5.13.

In the triangle case, let q a point in the annulus, be the circum-center of T with visible-radius vr. We know that the $\rho \leq d(p, q) \leq 2R_M(q) \leq C_{5.7} \cdot vr$ by Lemma 5.7. By Lemma 5.9 we know that the ball $B(q, vr/\sqrt{2})$ is empty at the time we inserted q. Thus, if there are two such points q and q' in A, it follows that the distance from q to q' must be at least $\rho/(\sqrt{2}C)$. Therefore the ball of radius $\rho/(2\sqrt{2}C)$ around each such point is disjoint. It follows by area arguments that there are at most a constant number of such points. ∎

The last lemma gives us an upper bound on our running time. We can improve our analysis by giving a better bound on the indegree for points added by procedure INSERT-CIRCUM-CENTER. First observe that $\Gamma_M(p)$ is at most a constant for circum-centers of triangles. It is not true that $\Gamma_M(p)$ is a constant for segment midpoints. But on average it is.

LEMMA 4.4. *Let E be an edge in G, the input PSLG, then the average indegree in the graph $\mathcal{G}$ of the midpoints added to E is a constant.*

Proof. Let E be an input segment and let $P = p_1, \ldots, p_k$ be the points added to E during the life of the algorithm. Further let E_i be the segment with center p_i. We know by Theorem 4.1 that the indegree of each point p_i in $\mathcal{G}$ is at most $C_{4.1} \cdot \log \Gamma_{M_i}(p_i)$ where M_i is the mesh just after inserting p_i. By Lemma 5.7 we know that each ridge out of p_i can have length bounded by $C_{5.7}$. On the other hand, up to a constant $\operatorname{lfs}(p_i)$ does not change throughout the life of the algorithm. Thus $\Gamma_{M_i}(p_i) \leq C|E_i|/\operatorname{lfs}_G(p_i)$.

We next define a DAG $\mathcal{H}$ with vertices P. We add a directed edge from p_i to p_j if p_i is an endpoint of E_j. Observe that the indegree is at most two.

CLAIM 4.1. *The out degree in $\mathcal{H}$ of p_i is bounded below by $C \log \frac{|E_i|}{\operatorname{lfs}_G(p_i)}$ for some constant $C > 0$.*

The Claim follows by observing that when the graph G is finally refined $\mathbb{R}(p_i)$ must be approximately $\operatorname{lfs}(p_i)$.

Note that $\frac{|E_i|}{\operatorname{lfs}_G(p_i)}$ is a constant for nodes in $\mathcal{H}$ with out-degree zero, sinks. By a standard charging argument, each non-sink charges its indegree in $\mathcal{G}$ to its children in $\mathcal{H}$. Thus each node will only pay for its indegree in $\mathcal{H}$ which is at most two. ∎

5 Structural Properties

In this section we present a collection of results that describe the types of configuration that can occur during the run of the procedure DELAUNAY-REFINEMENT. These results will then be used in Section 4 to obtain our upper bounds.

We first observe that, by assuming that the input is a PSLG with boundary, and the boundary will not change during the refinement, and boundary segments may be split into smaller segments. This is true because our search will never cross an input segment.

Throughout this section we assume that G is PSLG with boundary, M a constrained Delaunay triangulation of the interior of G, α_0 is the cutoff for skinny triangles. Let D_{α_0} equal the diameter of the maximum diameter skinny triangle in M.

The first set of lemmas show that in the case when the point added comes from our priority queue in procedure DELAUNAY-REFINEMENT, the circum-radii of all triangles in a cavity or a tent are bounded.

LEMMA 5.1. *Let G be a PSLG with boundary, M a constrained Delaunay triangulation of the interior of G, and p a point of M.*

$$R_M(p) = \max\{\, vr(T) \mid T \in \mathrm{Star}(p)\},$$

where $vr(T)$ is the visible-radius of the triangle T.

Proof. The proof follows by observing that the maximal empty circles containing p with respect to the visible-radius metric are precisely the circum-circles of the triangles in $Star(p)$ ∎

We next need to bound the number of fat triangles in any cavity or tent.

LEMMA 5.2. *Let p be a point not in M. Then the number of fat Delaunay triangles in the triangulation of the cavity before and after insertion of p is at most $\frac{2\pi}{\alpha_0} - 2$ and $\frac{2\pi}{\alpha_0}$, respectively*

Proof. See full paper. ∎

LEMMA 5.3. *If triangles T_1 and T_2 share an edge e and T_1 is fat then*

$$\frac{\mathrm{vr(T_1)}}{\mathrm{vr(T_2)}} \leq \frac{\mathrm{2vr(T_1)}}{\mathrm{dia(T_2)}} \leq \frac{\mathrm{2rad(T_1)}}{\mathrm{dia(T_2)}} \leq \frac{1}{\sin\alpha_0},$$

Proof. See full paper.

We next show that the maximum skinny triangles have small visible-radius.

LEMMA 5.4. *Let T be maximum diameter skinny triangle then* $\mathrm{vr(T)} \leq \mathrm{C_{5.4}dia(T)}$ *where the constant $C_{5.4}$ only depends on α_0.*

Proof. By Lemma 4.2 all the triangles in M from T to either the circum-center p or the edge which obscures p are fat and in the cavity of p. Let T_k be the triangle last triangle on the path. Consider the following chain of inequalities:

$$\frac{\mathrm{vr(T)}}{\mathrm{dia(T)}} \leq \frac{\mathrm{vr(T_k)}}{\mathrm{dia(T)}} \leq (1/\sin\alpha_0)^k$$

The last inequality follows from Lemma 5.3. By Lemma 5.2 we know that $k \leq \pi/\alpha_0 - 2$. ∎

We next show that if a triangle has almost maximum diameter then no skinny triangle has large visible-radius.

LEMMA 5.5. *If T is skinny triangle in M then* $\mathrm{vr(T)} \leq \mathrm{C_{5.5}D_{\alpha_0}}$

Proof. The lemma follows by observing that the visible radius is nondecreasing as we move up the chain of obtuse triangles. Thus we may assume that T is a maximal skinny triangle and simply apply Lemma 5.4. ∎

LEMMA 5.6. *If S is a simplex,* $\mathrm{dia(S)} \geq \alpha\mathrm{D_{\alpha_0}}$*, and S encroaches on a triangle T then* $\mathrm{vr(T)} \leq \mathrm{C_{5.6}}\alpha\mathrm{dia(S)}$

Proof. If T is skinny, we are done by Lemma 5.5. We may assume that T is fat. By Lemma 5.2 we know that number of fat triangles in the cavity is bounded and by Lemma 5.3 their visible radii can only grow by a constant for each fat triangle. ∎

LEMMA 5.7. *Let S be a simplex in M,* $\mathrm{dia(S)} \geq \alpha\mathrm{D_{\alpha_0}}$*, and p the circum-center of S. If E is an edge from p to q adding p to M then* $|p-q| \leq 2C_{5.7}\alpha\mathrm{dia(S)}$

Proof. Let T be a triangle in the cavity of p in M. Now $|p-q| \leq 2\mathrm{vr(T)} \leq 2\mathrm{C_{5.6}}\alpha\mathrm{dia(S)}$ ∎

5.1 Parents in an Annulus In this subsection and the next subsection we show that the number of Steiner points in an annulus of radius from r to $2r$ that have an edge to some earlier generated point, is at most a constant. This will be crucial in showing an upper bound on the work performed.

Define the following directed graph $\mathcal{G}$ on the set of all output points procedure DELAUNAY-REFINEMENT: Add a directed edge, called a **ridge**, from each Steiner vertex p in the output mesh to each of the vertices that appear on the boundary of the cavity at the time immediately after inserting the vertex p. By definition, no edges in our graph $\mathcal{G}$ emanate out of vertices of the initial mesh, but edges can point to them. Also note that an edge can only travel in the reverse direction of time, i.e. it can only connect a Steiner vertex to a vertex that already existed in the mesh at the time of inserting the Steiner vertex. This means that there are no cycles in our graph $\mathcal{G}$. The number of directed edges in our graph $\mathcal{G}$ is precisely the total number of triangles created during Ruppert's refinement algorithm, excluding the triangles created for the purpose of computing initial constrained Delaunay triangulation. On the other hand, the number of directed edges equals the sum of in-degrees of all vertices in the graph $\mathcal{G}$. We will now provide an upper bound to the in-degrees of a vertex, which will later be used to establish the upper bound for the run-time of the algorithm.

We first give an upper and lower bound on the length of the edge into a point.

LEMMA 5.8. *Let p be a point in the output mesh produced by procedure DELAUNAY-REFINEMENT(G) and q be a point with an edge to p in the graph $\mathcal{G}$ then*

$$C_{5.8}\,\mathrm{lfs}_G(p) \leq \mathrm{lfs}_M(p) \leq dist(q,p) \leq 2R_{M'}(p) \leq 2R_G(p),$$

where M is the output mesh, M' is the mesh just prior to adding q, and $C_{5.8}$ is some constant determine by a mesh size analysis of Delaunay refinement.

Proof. The first inequality comes from Ruppert's original analysis of Delaunay refinement [Rup95]. There have been many analyses with better bounds that have appeared since [She00, MPW02]. Since we have slightly changed the algorithm we have included a proof of this inequality in Section 6.

To see the second inequality, first note that q has to be a Steiner vertex. Clearly the interior of the disk centered at p of size $\mathrm{lfs}_M(p)$ is free of other vertices, including q.

Let us now establish the third bound. At the time of insertion of Steiner vertex q, the point p is on the boundary of the cavity of q. Hence, p is a vertex of some triangle T that disappears from the triangulation as a result of adding the Steiner vertex q. Let C be the circum-circle of T. Since q is visible to p the distance from p to q must be at most the diameter of the visible part of the interior of C to p. This diameter is at most $2vr_p(\mathrm{C})$. Thus $dist(p,q) \leq R_{M'}(p)$.

To see the last inequality simply observe that the radial spacing function is a non-increasing function. As we add points to G, we can only decrease the radial spacing function. ∎

The next lemma shows that when we add the circum-center p of a triangle to the mesh and p does not yield to an encroached edge then there is a large empty ball around the point p.

LEMMA 5.9. *Let p be the circum-center of a triangle T with circum-radius r that is added to the mesh by procedure DELAUNAY-REFINEMENT(G) then the following are true:*

1. *No edge of G intersects the ball $B(p, r/\sqrt{2})$.*
2. *There is no mesh point in the ball $B(p, r/\sqrt{2})$ at the time that p is added.*

Proof. To prove the first claim suppose the that some edge segment e of G intersects the ball $B(p, r/\sqrt{2})$. Since p did not yield to the segment e, the midpoint of e must not belong to the ball $B(p, r)$. It follows that p must encroach on one of the two subsegments of e. Thus p should have yielded to e, a contradiction.

The second claim follows by observing that all of $B(p, r/\sqrt{2})$ is visible to p and thus there can be no mesh point in $B(p, r/\sqrt{2})$. ∎

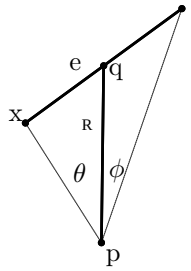

Figure 2: A Slanted-Tee.

It is not true that the midpoint of a segment added by procedure DELAUNAY-REFINEMENT(G) will have a large empty ball around it at the time it was added. In the next subsection we address this issue.

5.2 Slanted-Tees and Their Critical Angle In this section we develop definitions and properties we will need to bound the work performed by inserting midpoints of edges.

Throughout let e be a segment of an input edge and q its midpoint. Suppose that, on adding q to the mesh, we introduce a ridge R from q to a point p.

DEFINITION 5.1. *The pair consisting of the segment e and the ridge R is called a* **slanted-tee***, with* **base** *p and* **midpoint** *q. Let x be the endpoint of e that is closer to p, breaking ties arbitrarily. We call the triangle $x-q-p$ the* **critical triangle** *and the angle $x-p-q$ the* **critical angle***. The triangle $q-y-p$ is the* **acute triangle***. See Figure 2.*

We will need a simple technical lemma about a triangle.

LEMMA 5.10. *Let A, B, C be the angles and a, b, c be the opposite sides of a triangle and $0 < \alpha$ be some fixed constant. If $|a| \geq \alpha|c|$ and $B \geq \pi/2$ then there exists constants $0 < \beta$ and $1 < \gamma$ such that either $A \geq \beta$ or $\gamma|c| \leq |b|$.*

Proof. See full paper. ∎

DEFINITION 5.2. *We say a slanted-tee is skinny if its critical angle θ satisfies $\sin\theta \leq 1/(2\sqrt{2}C_{5.7})$. Otherwise it is fat. Thus a tee is skinny if the circum-radius r of the critical triangle satisfies $\sqrt{2}|R| \leq r$.*

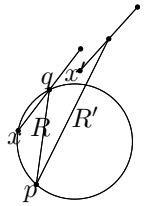

Figure 3: Two skinny Tees facing to the right.

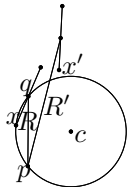

Figure 4: Two skinny Tees facing each other

.

We next show that there is at most a constant number of slanted-tees with midpoints in a constant ratio annulus with ridges to a point p, the center of the annulus. We first need a technical lemma about the distance between slanted-tees.

We say a tee is **right-facing** if the critical triangle is to the left of the ridge otherwise we say it is **left-facing**. We say two tees **face(away from)** each other if the angle between them is $< \pi$ and the left tees is left(right) facing and the right is right(left) facing.

LEMMA 5.11. *If T and T' are two fat slanted-tees with ridges R and R' that do not face each other then either the angle formed by R and R' is at least $C_{5.11}$, $|R| \geq C'_{5.11}|R'|$, or $|R'| \geq C'_{5.11}|R|$ for constants $C_{5.11} > 0$ and $C'_{5.11} > 1$.*

Proof. It will suffice to handle the case when T' is right-facing and to the right of T. We are done if T is left-facing. Thus we may assume that T is also right-facing. If the midpoint q of T is not in the critical triangle of T' then the angle between R and R' must be at least $C_{5.2}$. On the other hand, if q is in the critical triangle of T' then the distance from p to q' is at least the distance from p to y, where y is the far point on e and the angle between R and R' is bounded below by the acute angle of T. Applying Lemma 5.10 to the acute triangle of T, we are done. ∎

LEMMA 5.12. *If T and T' are two skinny slanted-tees with ridges R and R' that do not face away from each other then one of the following hold 1) there is an input edge between R and R', 2) the angle formed by R and R' is at least $C_{5.12}$ or 3) $|R| \geq C'_{5.12}|R'|$ or $|R'| \geq C'_{5.12}|R|$ for constants $C_{5.12} > 0$ and $C'_{5.12} > 1$.*

Proof. We first consider the case that T and T' are both right-facing. We may assume that R is to the left of R'. If there is an input edge between R and R' we are done. Otherwise we assume there is none. Since there is no input edge between R and R' it follows that the point x', see Figure 3, cannot be interior to the circum-circle of critical triangle of T. Thus either the angle formed by R and R' is big or $|R| \leq C|R'|$ for some $C > 1$.

Finally we consider the case where T and T' are facing each other. See full paper and Figure 4. ∎

LEMMA 5.13. *If $A(p, \rho, 2\rho)$ is an annulus centered at p then the number of slanted-tees with base p and midpoint in A is at most a constant.*

Proof. We count the fat slanted-tees separately from the skinny ones.

We first consider the fat tees. If we write R for a right-facing tee and L for a left-facing one then all the tees in the annulus form a cyclically ordered sequence of R's and L's. We partition the annulus into a constant number of annuli of ratio $C_{5.11}$ and count the number in each annuli separately.

By Lemma 5.11 and the fact that the ratio between the outer radius and inner radius is at most $C_{5.11}$, the angle between two tees is bounded below, except if the tees face each other. But the number of consecutive tees that face each other is just the number of consecutive tees that face away from each other, which is a constant.

The case for skinny tees is symmetric to the fat tee case. ∎

6 Output Size

The proof that our modifications to known Delaunay refinement algorithms does not effect the size optimality can be found in the full paper. The proof uses relatively standard and known techniques. The only change is to replace the nearest neighbor function with a nearest visible neighbor function.

7 Conclusion and Open Questions

It is interesting that several very simple incremental algorithm specifications in computational geometry have run times that are not known. On the positive side, it is well known how to construct an algorithm for inserting n points into a Delaunay triangulation in random order gives an expected run time of $O(n \log n)$ when we start with a fixed constant-size initial triangulation [dBvKOS00]. It is open how to construct a provably efficient incremental algorithm when we start with a large initial configuration? Our Delaunay refinement is a more constrained version of the problem since we can only add the circum-centers of the triangles that are present at the time. As we add these circum-

centers other circum-centers become available for insertion, where as the corresponding insertion problems for sorting are much easier and well studied [CLRS01].

8 Acknowledgments

I would like to thank Hal Burch and Noel Walkington for their contribution to an earlier manuscript which considered the case of inputs consisting of points only and Jernej Barbic for his $\Omega(n^2)$ example of a bad insertion order as well as enumerable discussions.

References

[Bar02] Jernej Barbic. Quadratic example for delaunay refinement. 2002.

[BE92] Marshall Wayne Bern and David Eppstein. Mesh generation and optimal triangulation. In Ding-Zhu Du and Frank Kwang-Ming Hwang, editors, *Computing in Euclidean Geometry*, number 1 in Lecture Notes Series on Computing, pages 23–90. World Scientific, 1992.

[BEG94] Marshall Bern, David Eppstein, and John Gilbert. Provably good mesh generation. *J. Comput. System Sci.*, 48(3):384–409, 1994. 31st Annual Symposium on Foundations of Computer Science (FOCS) (St. Louis, MO, 1990).

[BET99] Marshall W. Bern, David Eppstein, and Shang-Hua Teng. Parallel construction of quadtrees and quality triangulations. *International Journal of Computational Geometry and Applications*, 9(6):517–532, 1999.

[BOG01] Charles Boivin and Carl F. Ollivier-Gooch. Guaranteed-quality simplicial mesh generation with cell size and grading control. *Engineering with Computers*, 17(3):269–286, 2001.

[Che87] P. Chew. Constrained Delauany triangulation. In *in Proc. ACM Symposium on Comp. Geometry*, pages 213–222, 1987.

[Che89] L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report 89-983, Computer Science Department, Cornell University, 1989.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT-press and McGraw-Hill, 2 edition, 2001.

[dBvKOS00] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational geometry*. Springer-Verlag, Berlin, revised edition, 2000. Algorithms and applications.

[Ede01] H. Edelsbrunner. *Geometry and Topology of Mesh Generation*. Cambridge Univ. Press, England, 2001.

[Kad01] Clemens Kadow. A fully incremental Delaunay refinement algorithm. Posterpresentation 10th International Meshing Roundtable, October 2001.

[KM92] T. C. Kau and David M. Mount. Incremental construction and dynamic maintenance of constrained delaunay triangulations. In *Proc. 4th Canad. Conf. Computational Geometry*, pages 170–175, 1992.

[MPW02] Gary L. Miller, Steven E. Pav, and Noel J. Walkington. A finer analysis of the Delaunay Refinement Algorithm. in preparation, november 2002.

[Rup95] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995. Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (Austin, TX, 1993).

[Sei88] R. Seidel. Constrained Delauany triangulation and voronoi diagrams with obstacles. Technical Report Rep 260, IIG-TU Graz, 1988.

[She95] J. R. Shewchuck. Triangle: A two-dimensional quality mesh generator and Delaunay triangulator. 1995.

[She96] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[She97] Jonathan Richard Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1997. Available as Technical Report CMU-CS-97-137.

[She00] Jonathan Richard Shewchuk. Mesh generation for domains with small angles. In *Proceedings of the Sixteenth Annual Symposium on Computational Geometry (Hong Kong, 2000)*, pages 1–10 (electronic), New York, 2000. ACM.

[She02a] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1-3):21–74, May 2002.

[She02b] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Comput. Geom.*, 22(1-3):21–74, 2002. 16th ACM Symposium on Computational Geometry (Hong Kong, 2000).

[STÜ02] Daniel Spielman, Shang-Hua Teng, and Alper Üngör. Parallel delaunay refinement:algorithms and analyses. In *Proceedings, 11th International Meshing Roundtable*, pages 205–218. Sandia National Laboratories, September 15-18 2002.

[WS87] C. Wang and L. Schubert. An optimal algorithm for constructing the Delauany triangulation of a set of line segments. In *in Proc. ACM Symposium on Comp. Geometry*, pages 223–232, 1987.