Incremental Construction and Dynamic Maintenance of Constrained Delaunay Triangulations

Thomas C. Kao David M. Mount
Department of Computer Science
University of Maryland
College Park, MD 20742
Email: kao@cs.umd.edu (Internet)

Abstract

The Delaunay triangulation $DT(V)$ (for a point set V) is an important geometric structure. Sometimes for many applications it is desirable to add constraints requiring that a certain set E of edges be present in the triangulation. This is called a ***constrained Delaunay triangulation***, for a point set V, denoted as DT(V, E). An algorithm is presented for the dynamic maintenance of constrained Delaunay triangulations in the plane. There are four major operations for manipulating the triangulation: point insertion, point deletion, constraint chord insertion and constraint chord deletion. This is the first algorithm to implement all four operations under one consistent framework.

A randomized algorithm is presented, so that under fairly weak assumptions on the positions of the N points in the set V, an arbitrary set E of M nonintersecting chord deletions can be performed in $O(N \log M)$ expected time, and a set of M chord insertions can be performed in $O(N \log N \log M)$ expected time.

1 Introduction

The triangulation of a point set V is obtained by joining the points of V by nonintersecting straight line segments so that every region internal to the convex hull of V is a triangle. The ***Delaunay triangulation*** of the point set V, denoted as $DT(V)$, is a triangulation of V with the additional property that the circumcircle of any triangle in the triangulation contains no points of V in its interior. This is also called the "empty-circle" criterion [9]. In the plane it is known that the Delaunay triangulation is the triangulation that maximizes the smallest angle over all possible triangulations of the input point set V. This is also called the ***max-min angle*** criterion [11].

Although the Delaunay triangulation is widely used for various purposes, it is common to impose additional chord constraints for practical considerations. The resulting triangulation is called a ***constrained Delaunay triangulation***, denoted as DT(V, E) for a point set V and a set E of edges between vertices in V. Lee and Lin [11] defined it using the "truncated empty circles" (see Figure 1) such that the circumcircles are truncated (or cut) by the constraint chords. More specifically, a triangle $t = \triangle(u, v, w)$ is empty if the circumcircle of t contains no vertices ***visible*** from all three vertices u, v, w. Those constraint chords are edges of the planar straight line graph (PSLG). Alternatively, the constrained DT(G) of a PSLG is a triangulation of the graph $G = (V, E)$ that maximizes the minimum angle over all possible triangulations of the graph [11]. We will use planar subdivision and PSLG interchangeably. We also denote constraint edges (or segments) as **constraint chords** or simply **chords**.

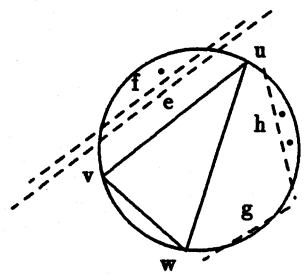

Figure 1: A truncated circle

There are quite a few applications of constrained DTs. For example, mesh generators for finite element analysis [2] [3] may require a (Delaunay) triangulation of the points subjected to given boundary constraints. In geographic data processing it is often desirable to preserve natural features like ridges or valleys [7] [14].

An algorithm is presented for the dynamic maintenance of constrained Delaunay triangulations in the plane. There are four major operations for manipulating the triangulation: point insertion, point deletion, constraint chord insertion and constraint chord dele-

tion. We show that the running time is proportional to the number of structural changes (k) for chord deletion ($O(k)$) and is $O(k \log k)$ for chord insertion.

Throughout the entire paper, assume that we are given a set V of N vertices in the plane, and a set E of M nonintersecting edges (chords) between vertices in V. By Euler's formula, we have $M \leq 3N - 6$. It is shown that under fairly weak assumptions on the positions of the N points in the set V, an arbitrary set E of M nonintersecting chord deletions can be performed in $O(N \log M)$ expected time, and an arbitrary set of chord insertions can be performed in $O(N \log N \log M)$ expected time by invoking our algorithms as a random permutation of the chords. This provides a simple $O(N \log N)$ expected time algorithm for computing DT(V, E) by first constructing an arbitrary triangulation AT(V, E) of points in V containing the given set E of nonintersecting constraint chords. Then in random order deleting each edge $e \notin E$ from this triangulation AT(V, E) to transform it into DT(V, E).

Previous Work: Lee and Lin [11] described an $O(N^2)$ algorithm for computing the constrainted $DT(V, E)$ of N vertices and M constraint chords. If the vertices of V formed a simple polygon, and the edge set E is the boundary of the polygon, the $DT(V, E)$ can be computed in $O(N \log N)$ time by divide-and-conquer. More details will be described Section 2.

Chew proposed the first $O(N \log N)$ optimal time algorithm for building the DT(V, E) [6]. By way of divide-and-conquer, the planar graph $G = (V, E)$ of N vertices and M constraint chords is subdivided into vertical strips such that there is exactly one endpoint in each strip. The DT of each strip is computed and then recursively merged together to form the final DT(G) of the entire graph.

De Floriani and Puppo [7] adopted a simple $O(k^2)$ time algorithm for the insertion of a constraint chord e by performing edge-swaps locally [13], where $k = L(e) + R(e)$. Therefore, the worst-case time complexity of their algorithm is $O(MN^2)$. If those M chords are introduced at the same time, the worst case running time is reduced to $O(N^2)$ [7].

2 Merging by Chord Deletions

In this section we present our algorithm for processing chord deletion in a constrained Delaunay triangulation. Before discussing our algorithm we mention a closely related problem: constrained DTs of simple polygons. An asymptotically optimal algorithm for computing the constrained Delaunay triangulation of any simple polygon is proposed by Lee and Lin [11]. They used the polygon-cutting theorem of Chazelle [4] and visibility polygon from a vertex [10] to compute constrained DT of a simple polygon of size k in $O(k \log k)$ time. They subdivided the input polygon Q into two roughly equal sized subpolygons $L(e)$ and $R(e)$, which are separated by a diagonal edge e of Q. The algorithm recursively constructs DT(L(e)) and DT(R(e)) for the two subpolygons. Then the two triangulations are "merged" to form DT(Q) in linear time. The overall time required for the construction of DT(Q) is $O(|Q| \log(|Q|))$.

In a nutshell, their algorithm achieved "merging" of two DTs separated by an edge e through the deletion of the constrained (diagonal) chord e. By removing the edge e, Lee and Lin were able to merge two DTs from two sub-polygons L(e) and R(e) in linear time.

Merging two Delaunay triangulations is a fundamental operation in Lee and Lin's construction. In fact, the first linear-time merging algorithm for DTs is proposed by Lee and Schachter [12]. They devised a procedure for merging two DTs separated by a straight line. Guibas and Stolfi [9] implemented a very similar merging scheme using their own invention, the **quad-edge** data structure.

The merging of two disjoint DTs separated by a line [12] can be considered as a special case of the more general merging of two DTs separated by an edge [11]. In general, we can generalize it further to consider the "stitching" of the Delaunay triangulation of any PSLG G by the deletion of any edge that does not belong to the final DT(G).

We will generalize the merging operation for PSLGs. Observe that merging two DTs separated by a chord is essentially the same as deleting a constraint chord from the triangulation. We are able to simplify Lee and Lin's algorithm for merging two DTs (by chord deletion) by eliminating the need to compute visibility polygons. In particular, given a chord $e = \overline{pq}$ to delete in the planar graph G, check if e belongs to the final DT(G). If it does, the merging is trivially done. Otherwise, e is deleted and the next task is to find a cross-edge $\overline{uv}$ that belongs to DT(G) such that $u \in L(e)$ and $v \in R(e)$ [11]. Lee and Lin chose v to be the vertex connected to both p and q in DT(R(e)), and find the vertex u to be on the smallest circle that passes through p, v and u, for all possible u's that are ***visible*** from the vertex v.

We can do better than that. Denote $deg(p)$ as the number of vertices adjacent to p in a graph. By choosing both u and v to be connected to p, we can find our first cross-edge $\overline{uv}$ that belongs to DT(G) in time proportional to $deg(p)$, and without the help of visibility polygon (from the point v).

The procedure (for finding the first cross-edge) is similar to Lee and Lin's algorithm for computing the Delaunay edges around a fixed vertex [11]. Let $L, L' \in L(e)$ and $R, R' \in R(e)$. We maintain the invariant that R', R, L, L' are four consecutive vertices counterclockwise around the fixed vertex P. We intend to compute a Delaunay edge from L to R, such that L, P and R forms a Delaunay triangle. We need to verify that the right hand side vertex R' is indeed outside the circle passing through L, P and R. If not, we delete the edge from P to R, let R be R' and repeat the process. We call this test *testR*. Similarly, we need to verify that the left hand side vertex L' is indeed outside the circle passing through L, P and R. If not, we delete the edge from P to L, let L be L' and repeat the process. We call this test *testL*. If both tests (testL and testR) are satisfied simultaneously, we have found the right answer.

Theorem 2.1 *Deleting a chord $\overline{pq}$ that results in k new edges in any constrained DT can be performed in $O(k)$ optimal time.*

3 Imposing Constraints by Chord Insertions

Before we can insert a constraint edge $e = \overline{pq}$, we assume that the endpoints p and q are already inserted into the current triangulation DT. There are two major steps required to insert edge e as a constraint edge of DT. First, we must identify the regions of the triangulation affected by the insertion of e. Any triangle intersected by e will not be in updated triangulation. Any triangle not intersected by e must be empty before the insertion of e and so is empty afterwards. Thus it suffices to consider the set of triangles intersected by e. Alternatively, we identify two polygonal chains $L(e)$ and $R(e)$ on the left and right side of the directed edge e from p to q. We call $L(e)$ and $R(e)$ (enclosed by e) as the two influence regions [7] associated with the insertion of edge e. For example, in Figure 2 the polygonal region $L(e)$ is formed by vertices p, q, r, s and t, and $R(e)$ is consisted of vertices q, p, u, v and w. If $e \in DT$ (e belongs to the final DT with other constraints), then the task is trivially done. Otherwise, the second step is to re-triangulate (using truncated empty-circle or max-min angle criteria) the two regions $L(e)$ and $R(e)$ respectively to form the constrained DT of the sites and the constraint edges. In the bottom half of Figure 2, we show the resulting DTs (for both $L(e)$ and $R(e)$) after the chord e is inserted.

Assume the indices of the endpoints (p and q) of e are given. Let $k = |L(e)| + |R(e)|$. The first step can be done in $O(k + deg(p))$ time by finding the triangle incident to p that intersect e and simply walking inside the current triangulation DT starting at vertex p. The second step requires more complex procedures to convert a simple polygon into its DT. For the insertion of any chord e, we can do it in $O(k \log k)$ time, using Lee and Lin's algorithm [11]. But because of the inherently complexities of Lee and Lin's algorithm (relying on the polygon-cutting theorem to triangulate, and the need to compute the visibility polygons for up to $N-3$ vertices), people (e.g. [7]) tend to use simpler but nonoptimal algorithms to construct DT.

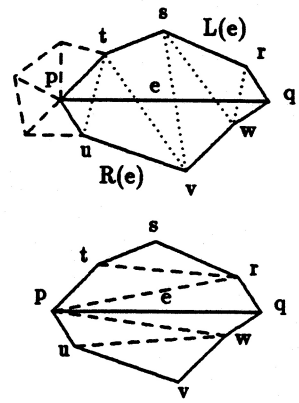

Figure 2: Constraint chord insertion

Our Approach: In the following discussion, we focus on the insertion of one single chord into the DT of N points. We are able to simplify the chord-insertion problem while preserving the $O(k \log k)$ time bound, where $k = |L(e)| + |R(e)|$. First of all, we use domain-specific knowledge to obtain greedy triangulations of the influence regions L(e) and $R(e)$) respectively, in linear time. Although any simple polygon can be triangulated in linear time by an optimal algorithm invented by Chazelle [5], the procedures involved are very complicated. As shown by many authors (e.g. De Floriani and Puppo [7]), both L(e) and R(e) are *edge-visible* polygons as seen from the edge e. We observe that any edge-visible polygon can be triangulated in linear time by using a simple greedy algorithm similar to the "Graham scan" for computing the convex hull of a simple polygon [8]. We take three consecutive vertices a, b, c at a time, and test if the vertex b is concave. If b is convex, we can safely cut-out the ear-triangle formed by a, b, c by adding the diagonal $\overline{ac}$ (and try the next triple of vertices). The correctness of the procedure follows from the fact that if the newly added edge $\overline{ac}$ intersects any part of the polygonal chain $L(e)$, either vertex a or c is not visible from the edge e, a contradiction. If vertex b is concave, the algorithm try next triple of vertices. Special care is taken to make sure the base edge e belongs to the last ear-triangle in the process.

After $L(e)$ is triangulated by the above procedure, we form the final DT by successively deleting each one of the $|L(e)| - 3$ chords (or diagonals) forming this initial (greedy) triangulation. The order in which these chords

are deleted is important for the efficiency of the procedure. Ideally chords should be deleted in a balanced fashion so that triangulations of roughly equal size are merged together on either side of the chord to ensure $O(|L(e)|\log(|L(e)|))$ time worst-case bound. The ordering can be obtained by depth first search of the dual-tree of the triangles of L(e) in the same time bound. We call the the ordering of diagonals as *centroid decomposition* of the dual tree, as it tries to break it in roughly two-halves. The same procedure is applied to the other edge-visible $R(e)$.

Theorem 3.1 *A chord $\overline{pq}$ can be inserted into a constrained DT in time $O(k\log k + deg(p))$, where k is the number of triangles intersected by the chord.*

4 Putting It All Together

4.1 Point Deletions Revisited

The first theoretically optimal algorithm for deleting a point from a planar Delaunay triangulation is proposed by Aggarwal, Guibas et al. [1]. They can compute a special kind of 3D convex hulls of k points in $O(k)$ time. The algorithm is intended to compute the DT of vertices of convex polygons in linear time. They also prove that the deletion of a point p from DT can be done in $O(k)$ time, where k is the number of vertices around p in the DT. The k vertices adjacent to the point p on a lifted 3D convex hull (paraboloid of revolution) [9] actually form a 3D *convex cone*, and therefore the deletion of p from the 3D convex hull can be done in $O(k)$ time. In the projected plane, these k vertices form a star-shaped polygon around p, and is denoted as $star(p)$. Any planar Delaunay triangulation is a projection of the lower half of the lifted 3D convex hull (paraboloid) [9]. Therefore, maintaining the DTs by deleting the point p (of degree k) takes $O(k)$ time.

Unfortunately, the $O(k)$ time algorithm can not be applied here. We observe that if there are constraint chords in the Delaunay triangulation, the projection from 3D convex hull does not quite work here. In fact, off-line incremental insertions of the k vertices of $star(p)$ does not work either, because it only gives the un-constrained version of the DT (of the points, without the edge constraints of $star(p)$). Fortunately, we can treat it almost like the chord-insertion problem. First, we compute a greedy triangulation of $star(p)$. This can be easily computed in $O(k)$ time by cutting out ears one by one. The rest is done by deleting all diagonals of the greedy triangulation, according to the ordering suggested by the polygon-cutting theorem [4]. Therefore, the time needed to delete a point from any DT is $O(k\log k)$.

Theorem 4.1 *Deleting a point of degree k from any constrained DT can be done in $O(k\log k)$ time.*

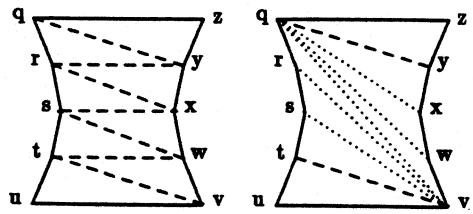

Figure 3: A worst case

4.2 Sequence of Chord Insertions or Deletions

Consider the following problem (P1):
Given the DT(V) of a set V of N points, we wish to insert a set E of M nonintersecting chords (between vertices in V) one by one in random order (for the purpose of computing DT(V,E)). What is the expected number of new triangles (or edges) destroyed in the process?
Equivalently, we can consider its dual problem (P2):
Given the DT(V,E) of a planar graph with $|V| = N$ vertices and $|E| = M$ chords, we wish to delete all the M chords one by one in random order (for the purpose of computing DT(V)). What is the expected number of triangles destroyed in the process?

A trivial upper bound to both problems P1 and P2 is $O(MN)$, as each chord can affect (or cut through) $O(N)$ triangles in the constrained DT. As M is bounded above by $3N$, the trivial bound is $O(N^2)$ structural changes. By structural changes we mean the number of triangles (or edges) destroyed (or created) in the process of M operations like inserting or deleting of chords.

We can show that in the worst case the $O(N^2)$ bound on structural changes is tight. For example, in Figure 3, $e_1 = \overline{qx}$, $e_2 = \overline{qw}$, $e_3 = \overline{qv}$, $e_4 = \overline{rv}$, and $e_5 = \overline{sv}$ are five chords to be inserted. If the order of chord insertions is e_1 followed by e_2, e_3, e_4 and finally e_5, the number of edges deleted (or structural changes) is $1+2+3 = 6$. In general, if there are n vertices on each side of the convex chain, n chord insertions of the same pattern can generate $1+2+\ldots+(n-2) = (n-2)(n-1)/2$ structural changes. On the other hand, if the main diagonal ($e_3 = \overline{qv}$ in the above figure) is inserted first, there will be exactly $2n-5$ structural changes.

Note that the same pattern in Figure 3 suggests that $O(n^2)$ edge-swapping locally [13] [7] is required in the worst case. On the other hand, because the $2n$ points form a simple polygon, only $O(n\log n)$ chord deletions are needed, as suggested by the polygon cutting theorem. Therefore, our chord-deletion strategy is superior to an edge-swapping one, at least in terms of the worst case performance for computing constrained DT of simple polygons.

Empirical studies suggest linear or close to linear performance for many kind of point distributions. By randomizing the sequence of insertions (or deletions), we hope to avoid such pessimistic bound like $O(N^2)$. In fact, we intend to show that, under random ordering of chord deletions (or insertions), the expected number of structural changes is no more than $O(N \log N)$.

We wish to extend the result to include chord insertions and deletions. We observe that every chord inserted into the DT(V) can cut through $O(N)$ triangles in the worst case. Equivalently, every chord deleted from the DT(V, E) can affect $O(N)$ vertices in the worst case. We wish to show that the deletions (or insertions) of a random sequence of $|E| = m$ nonintersecting chords from a DT(V, E) (with $|V| = n$) can be expected to generate $O(n \log m)$ structural changes.

We have to redefine the concept of so-called "general position" before we proceed with the main theorem (Theorem 4.4). In computational geometry, the typical *general position assumption* for planar Delaunay triangulation or Voronoi diagram is that "no four or more points are cocircular". We can generalize the concept by the following:

Definition 4.1 *Given three distinct points u, v, w (not collinear) in the plane, let $C(u, v, w)$ denote the circumcircle of these three points. a chord e is a* **cutting chord** *for triangle $\Delta(u, v, w)$ if the endpoints of e lie outside $C(u, v, w)$, e does not intersect the triangle $\Delta(u, v, w)$, and e intersects $C(u, v, w)$.*

Definition 4.2 *Consider a planar point set V and we are given two constants $c \geq 3$ and $0 \leq$ epsilon < 1. The point set V is said to be in* **(c,ϵ)-general position** *if for any circumcircle $C(u, v, w)$ (with radius r) of three points u, v, w in V, the annulus of disk C formed by radii r and $(1 - \epsilon)r$ contains at most c points.*

According the above definition, the "no four or more cocircular point assumption" is equivalent to the $(3, 0.0)$-general position assumption.

Definition 4.3 *Consider a circle C (with radius r) and a constant $0 \leq \epsilon < 1$. If the intersections between a (constraint) chord e and the disk C lie entirely within the annulus with radii r and $(1 - \epsilon)r$, the chord e is called a* **shallow cut** *for C. Otherwise, if e intersects with C and is not a shallow cut, it is called a* **deep cut** *for C.*

Let the triangle $t = \Delta(u, v, w)$. Because $t \in$ DT(V, E), there is no point p *visible* from u, v, w that is inside the circumcircle $C(u, v, w)$. By visible, we mean the ray from u (or v, w) to p does not intersect any constraint chord.

Definition 4.4 *The deletion of a constraint chord f can only affect the triangle $t = \Delta(u, v, w)$ only if f intersects $C(u, v, w)$, f is not* **blocked** *by any other constraint chord e as seen from all three vertices u, v, w, such that there is at least one point d invisible from u, v, w is hidden behind f. By saying a chord e blocks another chord (or cut) f, we mean the circular arc (closest to d) cut by e is a superset of the circular arc cut by f.*

If f is indeed blocked by e as seen from the triangle t, the deletion of either e or f will not affect t, unless both e and f are deleted. For example, in Figure 1, the deletion of chord h will affect the triangle $t = \Delta(u, v, w)$, while the deletion of the chord g is irrelevant to t. Only the deletion of *both* chords e and f will affect the triangle t. Now let us assume that there is an additional point d inside the circumcircle of t and is between chords e and f. Then the deletion of the chord e (before f) will cause the point d to be inserted into the triangle t, which will create two new triangles $\Delta(d, v, w)$ and $\Delta(d, w, u)$. On the other hand, the deletion of the chord f (before e) will not affect the triangle t. As far as the triangle $t = \Delta(u, v, w)$ is concerned, the presence of chord e will "negate" the effect of chord f, and therefore we do not consider blocked chords like f as relevant to $C(u, v, w)$. Only unblocked chords like e are considered relevant.

Lemma 4.2 *Given a positive constant $\epsilon < 1$, the number of nonintersecting deep cuts (that are not blocked by other cuts) for any circumcircle C is bounded above by $f(\epsilon) = \pi / \cos^{-1}(1 - \epsilon)$.*

Proof. Follows from simple trigonometry. □

Lemma 4.3 *Consider a DT(V, E) with $|V| = n$ points and $|E| = i$ nonintersecting constraint chords. Under the (c, ϵ)-general position assumption, the total number of triangles affected by those i constraint chords is bounded above by $(c + f(\epsilon)) \cdot 2n$, which is $O(n)$.*

Proof. Let $t(e)$ denote the number of triangles affected by the deletion of the constraint chord $e \in E$. The total number of triangles affected by those i chords is equal to the summation of $t(e)$ over all the i constraint chords. Note that every $t(e)$ is bounded above by $2n$, the number of triangles in any triangulation of n points.

Let $e(t)$ denote the number of constraint chords that affect the triangle $t \in$ DT(V, E). By a simple counting argument, we have

$$\sum_{t \in DT(V,E)} e(t) = \sum_{e \in E} t(e).$$

With the (c, ϵ)-general position assumption, we can show that $t(e)$ is bounded above by $c + f(\epsilon)$. First of all, the number of nonintersecting deep cuts for any circumcircle $C(u, v, w)$ is bounded above $f(\epsilon)$. Each such deep cut can hide zero or more points behind it, and therefore might affect the triangle $t = \Delta(u, v, w)$. Secondly, the number of shallow cuts for $C(u, v, w)$ is unbounded, but all the intersections with $C(u, v, w)$ are within the

annulus with radii r and $(1-\epsilon)r$. Fortunately, there are at most c points there in the annulus, each of which can be the hidden behind at most one constraint chord (or shallow cut) that is not blocked by other chords. Therefore, there can be at most $c + f(\epsilon)$ pairs of one constraint chord plus one hidden point. Summing over all possible triangle t, the total number of chords that affect triangles is bounded above by $(c + f(\epsilon)) \cdot 2n$. □

Theorem 4.4 *Consider the DT(V, E) of a point set V of n vertices and a set E of m nonintersecting chords. By deleting the m chords one by one in random order to compute DT(V), the expected number of structural changes (or number of triangles deleted) in the process is $O(n \log m)$, under the (c, ϵ)-general position assumption.*

Proof. We know that the total number of triangles affected by i chords at stage i is bounded above by $2(c+f(\epsilon))n$, independent of i or m. We assume the deletions of constraint chords run in stages. In stage i, there are i constraints chords left to be deleted. The whole process runs from stage m down to stage 0. For stage $i > 0$, the deletion of an "average" chord (from one of the i chords remaining) will affect at most $2(c + f(\epsilon))n/i$ triangles on the average. Therefore, the total number of triangles affected throughout the process is:

$$\sum_{i=1}^{m} \frac{2(c+f(\epsilon))n}{i} = O(n) \cdot \sum_{i=1}^{m} \frac{1}{i} = O(n \log m).$$

□

Corollary 4.5 *Given DT(V, E) with $|V| = N$ vertices and $|E| = M$ nonintersecting chords, the M chords can be deleted randomly one by one in $O(N \log M)$ expected time to compute DT(V), under the (c, ϵ)-general position assumption.*

Proof. With the help of Theorem 4.4, the total number of structural changes is expected to be $O(N \log M)$. Since the time to delete a chord with $k \leq O(N)$ structural changes is $O(k)$, the total time to delete all M chords is expected to be $O(N \log M)$. □

Corollary 4.6 *Given DT(V) with $|V| = N$ vertices and a set E of M nonintersecting chords between vertices in V, the M chords can be inserted randomly one by one in $O(N \log N \log M)$ expected time to compute DT(V, E), under the (c, ϵ)-general position assumption.*

Proof. With the help of Theorem 4.4, the total number of structural changes is expected to be $O(N \log M)$. Since the time to insert a chord with $k \leq O(N)$ structural changes is $O(k \log k)$, the total time to delete all M chords is expected to be $O(N \log N \log M)$. □

5 Bibliography

1. A. Aggarwal, L. J. Guibas, J. Saxe and P. W. Shor, "A Linear-Time Algorithm for Computing the Voronoi Diagram of a Convex Polygon", Discrete Comput. Geom., *4*, pp.591-604, 1989.
2. T. J. Baker, "Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation", Engineering with Computers, *5*, pp.161-175, 1989.
3. M. Bern, D. Eppstein and J. Gilbert, "Provably Good Mesh Generation", Proceedings 31st IEEE Annual Symp. on the Foundations of Computer Science, pp.231-241, 1990.
4. B. M. Chazelle, "A Theorem on Polygon Cutting with Applications", Proceedings 23rd IEEE Annual Symp. on the Foundations of Computer Science, pp.339-349, 1982.
5. B. M. Chazelle, "Triangulating a Simple Polygon in Linear Time", 31st Annual IEEE Symp. on Foundations of Computer Science, St. Louis, pp.220-230, 1990.
6. L. P. Chew, "Constrained Delaunay Triangulations", Algorithmica, *4*, pp.97-108, 1989.
7. L. De Floriani and E. Puppo, "An On-line Algorithm for Constrained Delaunay Triangulation", unpublished manuscripts, May 1991.
8. R. L. Graham, "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set" Inform. Process. Lett., *7*, pp.175-179, 1973.
9. L. Guibas and J. Stolfi, "Primitive for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", ACM Trans. Graphics, *4*, pp.74-123, April 1985.
10. D. T. Lee, "Visibility of a Simple Polygon", Computer Vision, Graphics, and Image Processing, *22*, pp.207-221, 1983.
11. D. T. Lee and A. K. Lin, "Generalized Delaunay Triangulation for Planar Graphs", Discrete Comput. Geom., *1*, pp.201-217, 1986.
12. D. T. Lee and B. J. Schachter, "Two Algorithms for Constructing a Delaunay Triangulation", Inter. J. Computer and Inform. Sci., *9*, 1980.
13. G. Sibson, "Locally Equiangular Triangulation", Computer Journal, *21*, pp.243-245, 1978.
14. L. Scarlatos and T. Pavlidis, "Hierarchical Triangulation Using Cartographic Coherence", CVGIP: Graphical Models and Image Processing, *54*, pp.147-161, 1992.