SERIE B — INFORMATIK

Fast Greedy Triangulation Algorithms [◇]

Matthew T. Dickerson*
Robert L. Scot Drysdale**
Scott A. McElfresh***
Emo Welzl[†]

B 94–09
April 1994

Abstract

The greedy triangulation of a set of n points in the plane is the triangulation obtained by starting with the empty set and repeatedly adding the shortest compatible edge, where a compatible edge is defined to be an edge that does not intersect any previously added edge. Computing this triangulation efficiently is a classical problem in computational geometry, and the triangulation is used in a number of applications. This paper presents a practical algorithm that computes the greedy triangulation of uniformly distributed points in expected time $O(n \log n)$ and a variant that is less dependent on the uniform distribution. In the process we give a method of testing the compatibility of an edge in $O(1)$ time and show that almost all edges in a greedy triangulation are either "short" or have both endpoints "near" the convex hull of the triangulation.

[◇]This paper benefitted from discussions with David Eppstein and others during the *MSI Workshop on Computational Geometry,* Stony Brook NY, October 23-24, 1992, with Cliff Stein and others in the Dartmouth Algorithms Seminar, with Lorenz Wernisch, and with Andrzej Lingas.
*This study was supported in part by the funds of the National Science Foundation, CCR-9301714. Department of Mathematics and Computer Science, Middlebury College, Middlebury VT 05753.
**This study was supported in part by the funds of the National Science Foundation, DDM-9015851, and by a Fulbright Foundation grant that helps support this author's sabbatical visit to the Freie Universität Berlin.
Dartmouth College, Hanover NH 03755, USA
***Department of Mathematics and Computer Science, Dartmouth College, Hanover NH 03755.
[†]Institut für Informatik, Fachbereich Mathematik und Informatik, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, Germany. This work was (partly) supported by the ESPRIT Basic Research Action No. 7141 (ALCOM II).

1 Introduction

1.1 Overview of the Results

The greedy triangulation (GT) of a set S of n points in the plane is the triangulation obtained by starting with the empty set and at each step adding the shortest compatible edge between two of the points, where a compatible edge is defined to be an edge that crosses none of the previously added edges. In this paper we present a simple, practical algorithm that computes the greedy triangulation in expected time $O(n \log n)$ and space $O(n)$ for points uniformly distributed over any convex shape. A variant of this algorithm should also be fast for many other distributions.

We first describe a surprisingly simple method for testing the compatibility of a candidate edge with edges in a partially constructed greedy triangulation. The new edge is tentatively added to the embedding of the partial GT and at most four constant time tests are done involving edges lying clockwise and counterclockwise from the candidate edge at each vertex. Even though there can be $O(n)$ edges adjacent to one of the endpoints, we are able to show that if we can determine where in angular order the new edge falls among a subset of at most 10 of those edges then we can perform the compatibility test and if necessary update the triangulation. Our method therefore provides a $\Theta(1)$ time edge test that requires only $\Theta(1)$ time to update the structure, $\Theta(n)$ time for initialization, and $\Theta(n)$ space. This compares favorably with the previous method of Gilbert [10], which requires $\Theta(\log n)$ time for an edge test, $\Theta(n \log n)$ time for an update, $\Theta(n^2 \log n)$ time for initialization, and $\Theta(n^2)$ space. It is also faster than the probabilistic edge pretest of Manacher and Zobrist [28], and it deterministically decides if a conflict exists rather than just finding a conflict with high probability.

We next prove that an edge cannot be greedy if a small disk centered at its midpoint contains a point from S in both half-disks. This fact allows us to prove a number of properties about the greedy triangulation for uniformly distributed points drawn from a convex compact region $\mathcal{C}$. We are able to prove that all edges in a greedy triangulation of uniformly distributed points are expected either to be short or to have both endpoints near the boundary of $\mathcal{C}$. Furthermore, we expect that only $O(n \log n)$ pairs of points are either short enough or have both endpoints close enough to the boundary of $\mathcal{C}$ to be in the greedy triangulation. Finally, we expect that only $O(n)$ pairs of points in S satisfy the condition that at least one of the two half-disks centered at the midpoint of the pair is empty. These lemmas also apply to the Delaunay triangulation.

This leads to the following algorithmic approach: generate the $O(n \log n)$ candidate edges that are short enough or whose endpoints are close enough to the boundary. Use an edge pretest based on empty half-disks to reject all but $O(n)$ of the candidates in constant expected time per candidate. Finally, sort these edges and attempt to insert them in order into the triangulation, using the fast edge compatability test.

This algorithm will run in $O(n \log n)$ expected time and will produce the greedy triangulation with very high probability for uniformly distributed points. We show how to modify it to create a two-phase algorithm that always computes the greedy triangulation. Its run time on uniformly distributed points is $O(n \log n)$ with very high probability. We also present another two-phase algorithm that is less tuned to uniform distributions. It always computes the greedy triangulation, and tries to balance the work done on short edges and longer ones. It runs in expected time $O(n \log^2 n)$ time on uniformly distributed points, and $O(n^2 \log n)$ in the worst case.

These algorithms should be compared to an algorithm by Levcopolous and Lingas [21]. For the more restricted case of points uniformly distributed in the unit square, their algorithm runs in expected time $O(n)$. Extending it to rectangles is straightforward. Extending it to non-rectangular convex shapes seems doable, but would require non-trivial modications of the algorithm and the analysis. Although their algorithm is beautiful theoretically, it has not been implemented and would be difficult to implement practically. Our algorithms not only work efficiently for more general compact convex regions, but the simplicity of our algorithms and smaller constant factors would make them preferable for most practical-sized problems.

1.2 Background

Efficiently computing the greedy triangulation is a problem of long standing, going back at least to 1970 [9]. A number of the properties of the GT have been discovered [19,23,27,28] and the greedy algorithm has been used in applications [5,28].

A straightforward approach to computing the GT is to compute all $\binom{n}{2}$ distances, sort them, and then build the GT an edge at a time by examining each pair in order of length and adding or discarding it based on its compatibility with the edges already added. It is easy to see that this method requires $O(n^2)$ space and time

$$T(n) = O(n^2 \log n + n^2 f(n) + n g(n)) \tag{1}$$

where $O(n^2 \log n)$ is the time required for an optimal comparison-based sort on $\binom{n}{2}$ distances, $f(n)$ is the time required to test new edges for compatibility, and $g(n)$ is the time required to update the data structure when a new greedy edge is added [30]. A naive test would compare each new potential edge to each of the existing edges (of which there are at most $O(n)$) for an $O(n^3)$ time algorithm. Gilbert [10] presented a data structure allowing an $O(\log n)$ time compatibility test and an $O(n \log n)$ time update, thus improving the algorithm's overall time complexity to $O(n^2 \log n)$, without adversely affecting space complexity. He does this by building a segment tree for each point in the set, where the endpoints of the "segments" are the polar angles between the given point and every other point in the set. Manacher and Zobrist [28] have since given an $O(n^2)$ expected time and $O(n)$ space greedy triangulation algorithm that makes use of a probabilistic method for *pretesting* compatibility of new edges. Note that our approach also uses this "generate and test" paradigm, and that we gain improvements over previous results by generating fewer edges and supplying more efficient tests.

Our approach can be viewed as an extension of Dickerson's [6]. He examined the idea of enumerating pairs of points in increasing order by distance, attempting to add them to the greedy triangulation, and quitting when the triangulation is complete. His hope was that only a small fraction of the $\binom{n}{2}$ edges would have to be examined. He showed that for points chosen uniformly from a disk only $O(n^{4/3})$ edges would be examined, but for points chosen uniformly from a polygon (or any shape with a flat side) $O(n^2)$ edges must be examined. The problem is long edges lying near the convex hull. This paper suggested using Gilbert's edge test, so because of intialization and update costs was not able to achieve an asymptotic speedup in the algorithm.

An alternate approach to "generate and test" is to generate only compatible edges. One way to do this was discovered independently by Goldman [11] and by Lingas [25]. The method uses the generalized or constrained Delaunay triangulation [4, 16, 35]. The constrained Delaunay triangulation is required to include a set of edges E. The rest of the edges in the triangulation have the property that the circumcircle of the vertices of any triangle contains no point visible from all three vertices.

This alternate approach computes the constrained Delaunay triangulation of the points with the current set of GT edges as the set E. The next edge to be added to the GT can be found in linear time from the constrained Delaunay triangulation. The triangulation must then be updated to include the new edge in E, which takes $O(n \log n)$ time in the worst case. This gives an $O(n^2 \log n)$ time and $O(n)$ space algorithm, thus improving the space complexity of Gilbert's algorithm without affecting the worst case time. Lingas [25] shows that his method runs in $O(n \log^{1.5} n)$ for points chosen uniformly from the unit square.

Recently Levcopoulos and Lingas, and independently Wang, have shown how to do the update step in $O(n)$ time, using a modification of the linear-time algorithm for computing the Voronoi diagram of a convex polygon [1], leading to an $O(n^2)$ time and $O(n)$ space algorithm in the worst case [20, 33]. More recently Levcopoulos and Lingas give a modification of this algorithm that is expected to take $O(n)$ time for points uniformly distributed in a square [21]. These methods are elegant, but are significantly more complicated to implement than our methods and should be slower for practical-sized problems.

One use of the greedy triangulation is as an approximation to the minimum weight triangulation (MWT). Given a set S of n points in the plane, a Minimum Weight Triangulation (MWT) of S is a triangulation that minimizes the total length of all edges in the triangulation. The MWT arises in numerical analysis [23, 26, 30]. In a method suggested by Yoeli [36] for numerical approximation of bivariate data, the MWT provides a good approximation of the sought-after function surface. Wang and Aggarwal use a minimum-weight triangulation in their algorithm to reconstruct surfaces from contours [34]. Though it has been shown how to compute the MWT in time $O(n^3)$ for the special case of n-vertex polygons [15], there are no known efficiently computable algorithms for the MWT in the general case [30]. We therefore seek efficiently computable *approximations* to the MWT.

Although neither the GT nor the Delaunay triangulation (DT) yields the MWT [27,26], the GT appears to be the better of the two at approximating it. In fact, for convex polygons the GT approximates the MWT to with a constant factor while the DT can be a factor of $\Omega(n)$ larger [19]. For general point sets, the DT can be a factor of $\Omega(n)$ larger than the MWT, but the best lower bound for the GT is $\Omega(\sqrt{n})$ [14,18]. For points lying on a convex polygon or uniformly distributed points in a square, both the GT and the DT are expected to be within a constant factor of the MWT [21,3]. For these reasons a large amount of effort has gone into finding efficient methods for computing the greedy triangulation.

Other heuristics for approximating the MWT have also been developed. Plaisted and Hong have developed a complicated polynomial-time heuristic that is guaranteed to be within a factor of $O(\log n)$ of the MWT [29]. The best heuristics known so far are those of Lingas [24] and Heath and Pemmaraju [12]. These approaches make use of the convex hull and a spanning tree to create a single cell, and then use an optimal cell triangulation algorithm generalized from Gilbert's dynamic programming approach for computing the MWT of points on a convex polygon [10]. Lingas' method begins with a minimum spanning tree derived from the convex hull and the DT, and provably produces a triangulation at least as good as the DT. The method of Heath and Pemmaraju begins with a spanning tree derived from the convex hull and the GT, and provably produces a triangulation always at least as good as the GT. In practice, both methods work extremely well, with that of Heath and Pemmaraju proving slightly better. However, though both methods appear empirically to produce triangulations much better than the GT, both of these algorithms require $O(n^3)$ time which is impractical for large point sets. In [12], the authors only report on data for sets of size 50 and smaller. We also note that with the method of Heath and Pemmaraju, the greedy triangulation still remains of interest as it is a substep in their algorithm.

1.3 Notation

Throughout the paper, we let $d(p,q)$ be the distance from point p to q using the standard Euclidean distance metric.

2 A New Edge Test Method for the GT

We now present our new method for testing the compatibility of edges in a greedy triangulation. We will first present some definitions, and then a new theorem stating a property of any pair of points that does *not* form an edge in the GT. Following the theorem, we will present the edge test method with a proof of its correctness and a complexity analysis of the run time required by each of the operations.

Definition 1 *In a straight line planar graph T, a* clockwise chain *from p_1 (hereafter written "CW chain from p_1") is a sequence of points $p_1, \ldots, p_k$ such that for* $1 \leq$

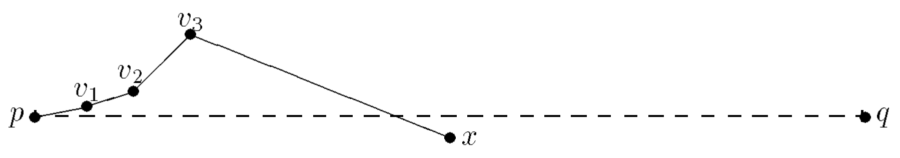

Figure 1: CCW chain intersecting segment pq

$i < k, p_i p_{i+1}$ are edges in T, and for $1 < i < k, p_i p_{i+1}$ is the next edge around point p_i in a clockwise direction from $p_{i-1}p_i$.

Definition 2 *Let $p_1, \ldots, p_k$ be a CW chain in straight line planar graph T. If (p_1, p_2) is the first edge in a clockwise direction from a segment $p_1 q$ (with $p_1 q$ not necessarily an edge in T), then we say that $p_1, \ldots, p_k$ is a* CW chain with respect to *$p_1 q$.*

We define a counter-clockwise chain, or CCW chain, in a similar fashion.

The new compatibility test method is based on the following lemma and theorem (see Figure 1).

Lemma 3 *Given a set S of n points, let x, y, and z be oriented so that they form a CW triangle. Let T be the partial triangulation in the standard greedy algorithm at the time when xz is tested for compatibility or at some later time. Let x, y, z be a CCW chain in T. Then triangle xyz contains no points in S in its interior and xz is either compatible with T or already in T. (The lemma is also true if CW and CCW are interchanged.)*

Proof: We first show by contradiction that there are no points in the interior of triangle xyz. Assume otherwise and let w be a first such point encountered by a line l parallel to xz passing through y as it sweeps across the triangle towards xz. Because xw is shorter than the longest side of xyz, it was considered as a possible edge. But it was not added, because the second edge in the CCW chain is yz, not yw. Therefore some edge must intersect its interior. But no such edge can exist, because it cannot cross xy or yz and no points interior to xyz lie on y's side of l when it reaches w. This contradiction shows that no points lie interior to triangle xyz.

This implies that no edge can block xz. Such an edge cannot cross xy or yz and cannot have an endpoint interior to triangle xyz. The only other possiblility would be for y to be an endpoint of the blocking edge, but then yz would not be the second edge in the CCW chain. Therefore nothing can block xz and it is compatible with T or already in T. $\square$

Theorem 4 (Clockwise/Counter-clockwise Chain Theorem) *Given a set S of n points and a non-negative integer $k \leq \binom{n}{2}$, let T be a partial greedy triangulation of S constructed using the standard greedy approach of examining the first k interpoint pairs in nondecreasing order of distance and adding to T exactly those edges compatible with previously added edges. Now let (p, q) be the $(k+1)^{st}$ pair to be tested with $\delta = d(p, q)$. If edge pq is not compatible with T, then T contains a CW or CCW chain $p, v_1, \ldots, v_k, x$ (or, respectively, $q, v_1, \ldots, v_k, x$) with respect to pq that has the following five properties:*

1) edge $v_k x$ intersects segment pq,

2) points $v_1, \ldots, v_k$ lie within the $\delta \times 2\delta$ rectangle R that is divided into two squares by segment pq and $d(p, v_i) < \delta$ (respectively $d(q, v_i) < \delta$) for $1 \leq i \leq k$,

3) if $d(q, v_1) \geq \delta$ (respectively $d(p, v_1) \geq \delta$) then $k = 1$,

4) the vertices $p, v_1, \ldots, v_k$ (all edges of the chain before $v_k x$) form a convex chain, and

5) $v_k x$ is the closest edge to p (respectively q) of all edges in T intersecting pq.

Proof: Assume edge pq is not compatible with T. Then by definition of compatibility, T already contains an edge intersecting pq. Furthermore, this edge is no longer than pq since it was already added to T in the "greedy" fashion.

(i) We first show that T contains an edge $v_k x$ with the properties that $v_k x$ is the closest edge to p (respectively q) of all edges intersecting pq, $d(p, v_k) < \delta$ (respectively, $d(q, v_k) < \delta$), and point v_k is inside the rectangle R.

Since pq is not a greedy edge, we know that there already exists in T an edge intersecting pq. Without loss of generality, let there be an edge in T intersecting pq at a point at least as close to p as to q. (If this is not the case, then simply reverse the roles of p and q for the remainder of the proof.) For ease of notation, rearrange the plane so that pq is the horizontal axis with p on the left. Let ab be the edge intersecting pq closest to p, calling the leftmost endpoint a. (If ab is vertical then call either endpoint a.) Note that a is at least as close to p as it is to q. There are three possibilities for the position of a: 1) a falls inside R and $d(p, a) < \delta$. In this case a satisfies the conditions for v_k, so we let $v_k = a$ (and $x = b$). 2) a falls inside of R but $d(p, a) \geq \delta$. Notice however that $d(q, a) \geq \delta$ since a is at least as close to p as to q. But b must fall in the circle of radius δ centered at v_k, and furthermore b is is on the other side of pq from a. By these constraints, b meets the requirements for v_k, and we let $v_k = b$. 3) Finally, a could fall outside of R. Because ab intersects pq, a must fall strictly to the left of R. But then b must lie in R, and the triangle inequality allows us to show that $d(p, b) < \delta$. Thus we can let $v_k = b$.

(ii) In part (i) we have described an edge meeting condition (5) in our theorem. We now show that there is CCW chain from p to x containing edge $v_k x$ and meeting conditions (2) through (4). (Since the chain contains $v_k x$ it automatically meets condition (1).) Label the point where $x v_k$ and pq intersect as o. Since $d(p, v_k) \leq \delta$ and $d(p, o) \leq \delta/2$ it is clear that for every point y in triangle $p o v_k$ (except possibly point v_k itself) we have $d(p, y) < \delta$. Furthermore, by our assumption xv_k was the closest greedy edge to p, and ov_k is a segment of this edge, and therefore there are no edges in T intersecting po or ov_k. We now show how to construct the CCW chain in a fashion similar to a Jarvis march [13]. Let $v_1 \in S$ be the point in triangle $\triangle pov_k$ that minimizes the counterclockwise angle of pv_1 with respect to pq. (If $\triangle pov_k$ is empty, then we have $v_1 = v_k$). We know that $d(p, v_1) \leq \delta$. Furthermore, there can be no greedy edges intersecting pv_1 since there are no points in $\triangle pov_k$ below ray $p\vec{v}_1$, and there are no edges intersecting po or ov_k. It follows that pv_1 is a greedy edge already added to T, and furthermore that pv_1 is the next CCW edge out of p with respect to pq. If $v_1 = v_k$ then we are done. Otherwise, we continue in the same fashion and choose the point v_2 that minimizes the angle v_1v_2. We see that v_1v_2 must also be an edge in T for the same reasons. We continue to choose the next point of minimal angle until we reach v_k. At this point, we have a CCW chain from p to v_k that meets condition (1) of the theorem. Furthermore, every point along this chain falls in $\triangle pov_k$ and therefore meets condition (2). ¿From the way in which these points were found, with each angle increasing with respect to the horizontal line, we see that the chain $p, v_1, \ldots, v_k$ is a convex chain and condition (4) is met also. Furthermore, by construction there are no points in the region bounded by this chain, po, and ov_k. Therefore $v_{k-1}v_k$ and v_kx must be adjacent to one another around v_k, making the entire chain a CCW chain.

We now prove by contradiction that our chain $p, v_1, \ldots, v_k, x$ meets condition (3) of our theorem as well. Assume that $k > 1$ and that point v_1 does not lie strictly inside the circle C of radius δ centered at q. (See Figure 2. In this figure, we show the boundary case where v_1 lies exactly on the circle C. For reference, the perpendicular bisectors of segments pv_1 and $v_{k-1}v_k$ are shown.)

Since p is on circle C and v_1 is on or outside the circle, we know that the perpendicular bisector of pv_1 passes through or over point q and that point x therefore lies closer to p than to v_1. However since this chain is convex, the same relationship holds true for all v_i and $v_{i+1}, 1 \leq i \leq k-1$. That is, we know that $d(v_{k-1}, x) < d(v_k, x)$. (The only way this could be false is if the convex chain turned by more than 270 degrees with respect to pq, but in this case either the chain would have to leave rectangle R or the edge v_kx would have to pass back through the chain, both of which are contradictions.) Given this and the fact that v_{k-1}, v_k, x forms a CCW chain, we can conclude from Lemma 3 that $v_{k-1}x$ must be an edge in the partial triangulation. Repeating this argument for v_{k-2} through v_1 shows that each must connect directly to x. Thus if v_1 lies outside of C and is the first edge in a CCW chain intersecting pq, then it must connect directly to x. $\square$

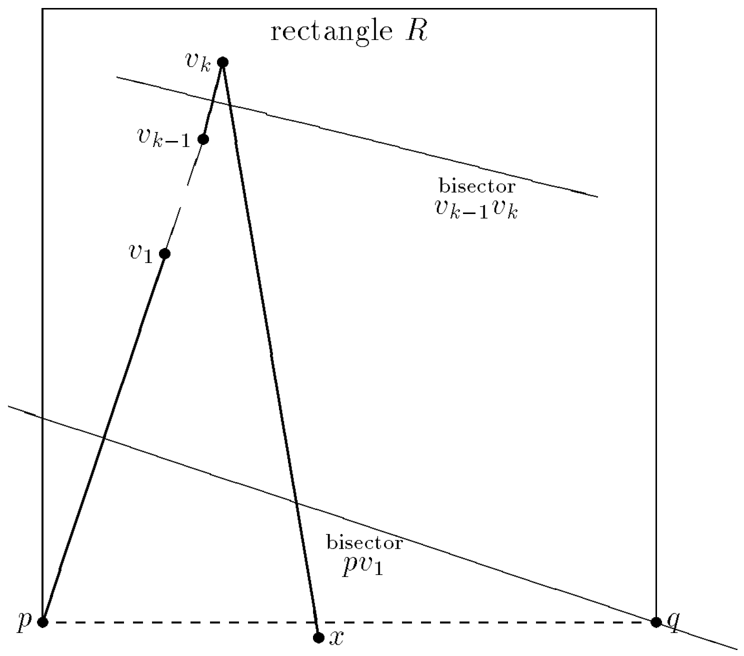

Figure 2: Condition (3). Proof by contradiction. (Assume point v_1 on or outside the circle C of radius δ centered at q. Here we show the boundary case with v_1 on C.)

2.1 Edge Test Method and Proof of Correctness

We now give a fast greedy triangulation edge test method based on Theorem 4. To determine whether edge pq is compatible, the algorithm examines the CW and CCW chains from p and q with respect to pq. The method actually requires that we examine at most only two edges on each of these chains.

2.1.1 Greedy Triangulation Edge Test for (p, q) in T

```
Step 1: CCW Chain from $p$
        1a. Find the next two points on the CCW chain from $p$ (with respect to $pq$).
              Label these points $v_1$ and $v_2$.
        1b. IF angle $\angle qpv_1 \geq \pi/2$ OR no CCW chain exists, THEN goto Step 2.
        1c. IF $v_1v_2$ intersects $pq$ THEN return FALSE ($pq$ is incompatible).
        1d. IF $v_2 = q$ THEN return TRUE
              (pair $(p, q)$ is compatible with $T$).
        1e. IF $v_1$ is strictly inside $C_q$ (the circle of radius $\delta$ centered at $q$)
              AND $v_2 \neq q$
                    THEN return FALSE.
                    ELSE goto Step 2.
```

```
Step 2: CW Chain from $p$
    2a. Find the next two points on the CW chain from $p$ (with respect to $pq$).
        Label these points $v_1$ and $v_2$.
    2b. IF angle $\angle v_1pq \geq \pi/2$ OR no CW chain exists, THEN goto Step 3.
    2c. IF $v_1v_2$ intersects $pq$ THEN return FALSE ($pq$ is incompatible).
    2d. IF $v_2 = q$ THEN return TRUE
        (pair $(p, q)$ is compatible with $T$).
    2e. IF $v_1$ is strictly inside $C_q$ (the circle of radius $\delta$ centered at $q$)
        AND $v_2 \neq q$
            THEN return FALSE.
            ELSE goto Step 3.

Step 3: CCW Chain from $q$
    3a. Find the next two points on the CCW chain from $q$ (with respect to $pq$).
        Label these points $v_1$ and $v_2$.
    3b. IF angle $\angle pqv_1 \geq \pi/2$ OR no CCW chain exists, THEN goto Step 4.
    3c. IF $v_1v_2$ intersects $pq$ THEN return FALSE ($pq$ is incompatible).
    3d. IF $v_2 = p$ THEN return TRUE
        (pair $(p, q)$ is compatible with $T$).
    3e. IF $v_1$ is strictly inside $C_p$ (the circle of radius $\delta$ centered at $p$)
        AND $v_2 \neq p$
            THEN return FALSE.
            ELSE goto Step 4.

Step 4: CW Chain from $q$
    4a. Find the next two points on the CW chain from $q$ (with respect to $pq$).
        Label these points $v_1$ and $v_2$.
    4b. IF angle $\angle v_1qp \geq \pi/2$ OR no CW chain exists, THEN return TRUE.
    4c. IF $v_1v_2$ intersects $pq$ THEN return FALSE ($pq$ is incompatible).
    4d. IF $v_2 = p$ THEN return TRUE
        (pair $(p, q)$ is compatible with $T$).
    4e. IF $v_1$ is strictly inside $C_p$ (the circle of radius $\delta$ centered at $p$)
        AND $v_2 \neq p$
            THEN return FALSE.
            ELSE return TRUE.
```

Proof of Correctness We now show that the above algorithm correctly determines whether edge pq is compatible with the edges already added to T (assuming the edges are added in greedy fashion). We give a proof for Step 1; the other three steps are exactly analagous.

The first two cases are obvious. First assume that v_1 is outside R (or equivalently that no CCW chain exists). It follows directly from Theorem 4 that if pq is not compatible we will find a CW or CCW chain on one of the subsequent steps and we can immediately go to Step 2. Now assume that v_1v_2 intersects pq. By definition, pq is not compatible with T and we can halt.

Now assume that $v_2 = q$. That is, the second edge on the CCW chain is v_1q. In this case Lemma 3 says that pq is compatible, so we can halt and answer TRUE.

If v_1 is strictly inside C_q and $v_2 \neq q$, then we claim that pq can not be compatible with T. Though we have not *yet* found an existing edge v_kx intersecting pq, we know that one exists, because assuming otherwise would lead to a contradiction. Suppose that pq were compatible and were added to the triangulation. Then q, p, v_1 would be a CCW chain, so by Lemma 3 triangle qpv_1 would contain no points in its interior and v_1q would be compatible. Because it is shorter than pq it would already be in the triangulation. But then v_1q would be the next edge CCW around $v1$ from pv_1, contradicting the assumption that $v_2 \neq q$. Therefore pq cannot be compatible.

There is one remaining case, which is when v_1 is inside rectangle R but not strictly inside circle C, $v_2 \neq q$, and v_1v_2 does not intersect pq. That this CCW chain will not lead to *any* edge intersecting pq follows directly from Condition (3) of Theorem 4.

We have now completed the proof of the correctness of our greedy edge test method. □

2.2 Implementation and Analysis

We now discuss the implementation of our edge test and analyze its efficiency. We store our greedy triangulation T as a graph using adjacency lists. Each adjacency list is represented as a circular linked list of edges ordered in polar (or rotational) order around the point. To find CW and CCW chains from a new edge we must determine where in rotational order the new edge would fit. That would normally take $O(n)$ time, or $O(\log n)$ if we stored the circular list in a binary tree. Fortunately, we can use properties of the greedy triangulation to do better.

Let p be a vertex in T. We consider the neighbors of p (in T) in CCW order. Let x and y be two consecutive neighbors, and let α be the CCW angle swept from px to py. We call the ordered pair (x, y) *closed* if $\alpha < \pi$ and xy is in T, and we call (x, y) *open* otherwise. (Note that if (x, y) is closed, by Lemma 3 the triangle pxy must be empty, so no new greedy edges can connect to p between px and py.) We call an edge an *open edge* if it connects p to a point in an open ordered pair. A *closed interval* is a wedge around p that is bounded by two adjacent open edges whose endpoints in CCW order do not form an open pair. (If the edges around p are viewed as spokes of a wheel, then the open pairs will correspond to pairs of spokes with no “rim” between them and the closed intervals will be maximal sections of the wheel with the entire “rim” present.) A *closed point* is a point incident to at least one edge but to no open edges.

To the each edge structure in the data structure for storing the edges as a circular linked list we add pointers to the next CW and CCW open edge and a flag to show whether the wedge lying between that edge and the next open edge CCW from it is an open ordered pair or a closed interval. These new fields will only be maintained for open edges, and will provide a doubly-linked circular list of open edges around each point.

Maintaining this structure when a new edge pq is added to T is fairly straightforward. We will discuss updates at p; q is symmetric. If pq is the first edge incident to p, the new edge will point to itself as an open ordered pair spanning 2π. If pq is added to the middle of an open pair (x, y) (note that new edges can only be added between edges of an open ordered pair), we must do two types of updates. First, (x, y) must be split into two pairs (x, q) and (q, y), which may be open or closed. To find out which, we must see if edges xq and yq are in T, which can be done in constant time. Second, we must see if the new edge closes off a previously open pair. To do that, we see if the CCW edge from pq has the same endpoint w as the CW edge from qp (and the symmetric case on the other side). In either case, we must be able to update the data structure so that a previously open interval (x, y) is now closed. To do this the new closed interval (x, y) must be merged with possible closed intervals lying to either side of it to form a single closed interval. All of these tests and updates can be done in constant time.

With this data structure, the edge test is done as follows. The new edge pq is located in the circular list of open edges at p and q. If pq falls in a closed interval for either point, then the edge is not compatible. Otherwise perform the test described in the previous subsection, using the open edges as the first edges in the CW and CCW chains. This test will take time proportional to the number of open edges.

Fortunately, the maximum number of open edges adjacent to any point p is 10. This is because any pair (x, y) with $\alpha < \pi/3$ must be closed. Because α is not the largest angle in triangle pxy, we know that xy is not the longest edge in the triangle. By Lemma 3, xy is in T, which means that (x, y) is closed. Therefore every open pair has $\alpha \geq \pi/3$, so so there cannot be more than six of them around p. If there are six, each edge is shared by two open intervals, so there are only six edges. If there are five or fewer open pairs, then there cannot be more than 10 open edges.

To initialize the data structure we simply create n empty circular lists in $O(n)$ time. Because there are $O(n)$ edges in the entire triangulation the total space required by all lists is $O(n)$.

Thus we have a data structure that requires $O(1)$ time for an edge test or an update, $O(n)$ time to initialize, and $O(n)$ space. For comparison, we end this subsection by noting that the method of Gilbert [10] requires $O(n^2 \log n)$ time for initialization, $O(n \log n)$ time for update, and $O(n^2)$ space.

3 A Necessary Condition for an Edge to be in the Greedy Triangulation

We begin by stating a simple and obvious lemma.

Lemma 5 (Convex Quadrilateral Rule) *Let pqr and pqs be two empty triangles in a greedy triangulation. If segment rs intersects segment pq (that is, $prqs$ is a convex quadrilateral), then $d(p,q) \leq d(r,s)$.*

Proof: The proof follows directly from the greedy method. Since both triangles $\triangle pqr$ and $\triangle pqs$ are empty, we know that no edge except pq intersects segment rs. Assume $d(r,s) < d(p,q)$. Then edge (r,s) would have been chosen first and added to the triangulation, and edge (p,q) would not have been added. $\square$

Heath and Pemmaraju [12] describe this as the property of "local optimality."

We now give an important (though less obvious) lemma[1] that states a necessary (but not sufficient) condition for an edge to be a greedy triangulation edge. This is a variant of Lemma 3.1 in [21].

Lemma 6 *Let p, q be a pair of points in a set S. Consider the disc D of radius $\delta = d(p,q)/(2\sqrt{5})$ centered at the midpoint of pq. Let pq divide the disc into two half-disks. If both half-disks contain at least one point in S then pq cannot be in the GT of S.*

Proof: For notational convenience, orient the plane so that segment pq is horizontal, with p on the left side. (For the remainder of the proof, refer to Figure 3.) Let a be the point closest to pq in the upper half-disk of D, and let b be the point closest to pq in the lower half-disk of D. Let C be the circle with pq as a diameter, and let α be the length of the shortest segment with endpoints on or outside of C that passes through D. Then $\alpha = \sqrt{d(p,q)^2 - 4\delta^2}$.

We will assume that pq is a GT edge and show that this leads to a contradiction. We will use the following observation.

Observation 1 Let p_1 and p_2 be any pair of points that are not connected by an edge in the GT. Then some GT edge intersecting segment p_1p_2 must have length $\leq d(p_1, p_2)$.

If pq is a GT edge, then ab cannot be a GT edge because they would intersect. Therefore, by Observation 1 there must be some GT edge of length less than $d(a,b)$ "cutting off" ab. This edge must be of length $\leq 2\delta$. We will show that no such edge exists either above or below pq.

[1]an earlier false version of this lemma with $\delta = d(p,q)/2$ was given in [6]. We give the corrected version here.

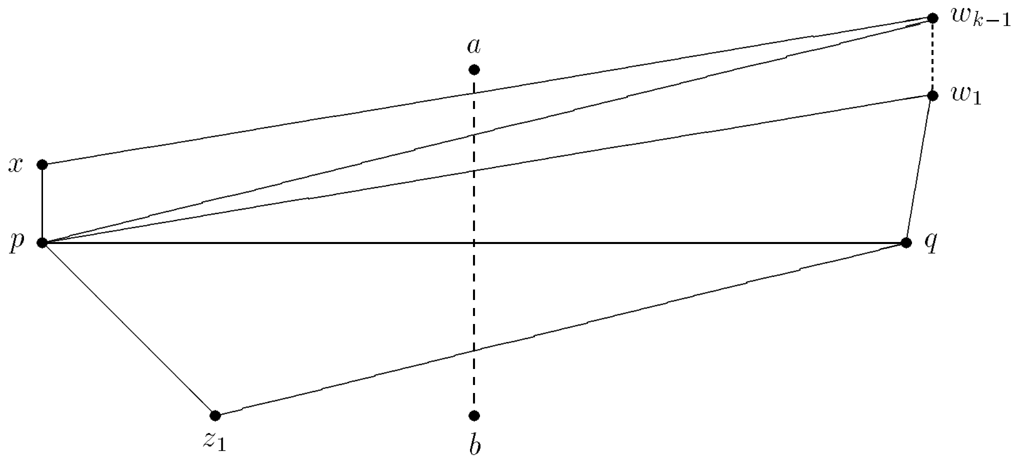

Figure 3: Edges intersecting ab

Since there are points on both sides of edge pq, it is not a CH edge, and therefore it must be an edge in two triangles – an upper and a lower. Let pw_1q be the upper triangle and let pz_1q be the lower triangle.

If w_1 and z_1 are both in C, then $d(w_1, z_1) < d(p, q)$ and thus by Lemma 5 pq could not be a GT edge and we have a contradiction. Without loss of generality, we therefore assume that w_1 is outside of C. Also without loss of generality, we assume that w_1 is closer to q than to p. That is, w_1 falls outside C on the q side. Since edge pw_1 cuts off ab and w_1 is outside C, it follows that $d(p, w_1) \geq \alpha$.

For notational convenience, we also label all the GT edges intersecting segment ab. The GT edges above pq that intersect ab we will label $e_1, \ldots, e_m$ and the edges below ab that intersect ab we label $f_1, \ldots, f_n$. More formally, consider the ray $\vec{ba}$. Leaving point b, it eventually passes through pq and then crosses a sequence of GT edges before reaching a. We label these edges in order $e_1, \ldots, e_m$. Likewise, ray $\vec{ab}$ leaves a and eventually crosses pq followed by a sequence of GT edges $f_1, \ldots, f_n$ before reaching b.

Our proof will be a case analysis on the edges $e_1, \ldots, e_m$ and $f_1, \ldots, f_n$. All of these edges cannot have both endpoints outside of or on C, because they comprise all of the edges crossing ab, and all of them would be longer than α. But our choice of δ causes α to be greater than 2δ, and by Observation 1, one intersecting GT edge must be shorter than $d(a, b)$. We will show that in fact some e_i above and some f_j below would have to have at least one endpoint in C, and we will then show that however this happens pq will not be a greedy edge.

Observation 2 At least one e_i and at least one f_j must have an endpoint in C.

Assume that none of the e_i had endpoints in C. Then all of the e_i have length at least α. Let f_l be the lowest-numbered f_j with an endpoint in C, and let y be an endpoint of f_l in C. Then all edges crossing ya are at least α long, but $d(y, a)$ must be shorter than α because y is in C and a is in D. This contradicts Observation 1. A symmetric argument shows that one of the f_j edges must also have an endpoint in C.

Observation 3 If e_k is the first e_i to have an endpoint in C, then $e_1, \dots, e_{k-1}$ are all of the form pw_i for $1 \leq i \leq k-1$, and e_k is of the form xw_{k-1}. Furthermore, x lies to the left of D. Finally, for some $1 \leq l \leq k-1$ there is an point w_l that lies within $d(p, q)$ of p. (See Figure 3 again.)

This observation follows from several points. First, no w_i can lie to the left of ray $q\vec{w}_1$ and closer than α to q unless a GT edge shorter than α intersects w_iq. But any edge e_i whose left endpoint lies to the left of C that intersects ab cannot have its right endpoint inside of C unless it lies closer than α to q. Furthermore, such an endpoint cannot lie to the right of ray $q\vec{w}_1$ without being outside of C. In either case the right endpoints of the e_i will be outside of C until some left endpoint lies in C.

Therefore we consider the first edge to have a left endpoint different than p. If this endpoint lies outside of C, then a similar argument to the one above will show that no left endpoints can lie within C either. Therefore no e_i will have an endpoint in C, contradicting Observation 2. This means that the first edge to have a left endpoint other than p will be e_k, and its left endpoint (which we will call x) must lie in C.

Finally, at least one of the edges pw_i for $1 \leq i \leq k-1$ must be shorter than $d(p, q)$. If it were not, then xq would cut off all of the edges $e_1, \dots, e_{k-1}$.

The Contradiction This structure is very constraining. We have constrained x to lie to the left of D and inside of C. Furthermore, if x were closer to q than α, then by Observation 1 one of the e_i with $1 \leq i \leq k-1$ would have to be shorter than α, which cannot be the case. (None of them have endpoints in C.) Therefore x is at least α from q. Finally, x must lie below the ray from q that is tangent to D from above.

We have also constrained point w_l to lie outside of C near q, but inside the circle of radius $d(p, q)$ centered at p. It must also lie below the ray from from p tangent to D from above.

Observation 2 says that some edge f_j will be the first edge below pq along ab with an endpoint y inside of C. But where can y lie? It must lie above either the ray from p tangent to D from below or the ray from q tangent to D from below. If it is in the left half of C, its distance to x will be less than α. If it is in the right half of C, its distance to w_l will be less than α. In either case we have a contradiction, because all intervening edges will be at least α long. (All of them have both endpoints on or outside of C.) $\square$

The proof above requires α to be larger than a number of different values. The strongest requirement arises in the last case, when we let y lie at the intersection of the ray from p tangent to D from below and C and we let w_l lie at the intersection of the ray from p tangent to D from above and the circle of radius $d(p,q)$ centered at p. The actual bound of $d(p,q)/(2\sqrt{5})$ is derived from a slightly worse configuration, which cannot be actually achieved. We require that the intersection points between the two rays from p tangent to D and the circle of radius $d(p,q)$ centered at p be at most α apart. This case is less than 5 percent worse than the actual worst case, and is less complicated to calculate and express.

We do not believe that our bound is tight. The worst example that we have been able to find is a diamond with pq as its diameter, with two additional points just outside of the midpoints of two opposite edges of the diamond. This case approaches the bound $\delta = d(p,q)/(2\sqrt{2})$ as closely as is desired.

4 An Analysis of the Edges in the Greedy Triangulation

For two points p and q in the plane, and for a real number $r > 0$, let $D(p,q,r)$ denote the closed disk of radius r centered at $(p+q)/2$. The line through p and q defines two closed semidisks of $D(p,q,r)$, denoted by $D'(p,q,r)$ and $D''(p,q,r)$ (without specifying which half is D' and D'', respectively).

Let us employ a constant $0 < \gamma \le 1$, fixed for the whole section. Given a set S of n points in the plane, we call a pair $\{p,q\}$ of points in S *plausible*, if $D'(p,q,r_1) \cap S = \emptyset$ or $D''(p,q,r_1) \cap S = \emptyset$, with $r_1 = \gamma|p-q|/2$. The previous section showed that if $\gamma = 1/\sqrt{5}$, only plausible edge edges can be in a greedy triangulation. When $\gamma = 1$ only plausible edges can be Delaunay, so the lemmas we prove about distributions of edge lengths apply to both greedy and Delaunay triangulations.

Our first goal is to estimate the expected number of plausible pairs in a set S of n points, uniformily distributed in a convex compact region $\mathcal{C}$. We normalize $\mathcal{C}$ to be of area 1 to simplify the notation.

Suppose p and q are points in $\mathcal{C}$, so that $D(p,q,r_1)$ is contained in $\mathcal{C}$ (r_1 as above). If we choose another $n-2$ points at random from $\mathcal{C}$, then the probability of $\{p,q\}$ being plausible is given by

$$\text{Prob}(D'(p,q,r_1) \text{ is empty or } D''(p,q,r_1) \text{ is empty}) \le$$

$$\text{Prob}(D' \text{ is empty}) + \text{Prob}(D'' \text{ is empty}) = 2\left(1 - \frac{r_1^2\pi}{2}\right)^{n-2} < 2e^{-r_1^2\pi(n-2)/2} .$$

We have to cope with pairs of points $\{p,q\}$ where $D(p,q,r_1)$ extends beyond the boundaries of $\mathcal{C}$: For $p \in \mathcal{C}$, let $\beta(p)$ denote the distance of p from the boundary of $\mathcal{C}$, or, in other words, the largest radius of a disk centered at p which is still contained in $\mathcal{C}$. Then we observe that for any pair if points p and q in $\mathcal{C}$, the disk $D(p,q,r_2)$, $r_2 = (\beta(p)+\beta(q))/2$, is contained in $\mathcal{C}$. It follows that for any pair $\{p,q\} \subset \mathcal{C}$ the probability of being plausible (after adding (n-2) random points) is bounded by

$$2e^{-r^2\pi(n-2)/2}, \qquad r = r_{p,q} := \min\{r_1, r_2\}\ ,$$

with r_1 and r_2 dependent on p and q as previously specified. Now let p and q be two random points in a random set of n points in $\mathcal{C}$. Then the probability of being plausible is bounded by

$$\int_{p\in\mathcal{C}}\int_{q\in\mathcal{C}} 2e^{-r^2\pi(n-2)/2}\,\mathrm{d}q\mathrm{d}p < \tag{2}$$
$$\int_{p\in\mathcal{C}}\int_{q\in\mathcal{C}} 2e^{-r_1^2\pi(n-2)/2}\,\mathrm{d}q\mathrm{d}p + \int_{p\in\mathcal{C}}\int_{q\in\mathcal{C}} 2e^{-r_2^2\pi(n-2)/2}\,\mathrm{d}q\mathrm{d}p\ .$$

In order to estimate the terms in (2) we use density functions $f_p(x)$, $p \in \mathcal{C}$, and $g(x)$. $\int_{-\infty}^{z} f_p(x)\mathrm{d}x$ is the probability for a random point in $\mathcal{C}$ to have distance at most z from p; clearly, $f_p(x) \le 2x\pi$ for all $p \in \mathcal{C}$ and $x \ge 0$. $\int_{-\infty}^{z} g(x)\mathrm{d}x$ is the probability of a random point in $\mathcal{C}$ to have distance at most z from the boundary of $\mathcal{C}$; here we have a bound of $g(x) \le U$, U is the perimeter of $\mathcal{C}$, for all $x \ge 0$.

Now the left term in (2) equals

$$\int_{p\in\mathcal{C}}\int_{x=0}^{\infty} 2f_p(x)e^{-x^2\gamma^2\pi(n-2)/8}\,\mathrm{d}x\mathrm{d}p \le \int_{x=0}^{\infty} 4x\pi e^{-x^2\gamma^2\pi(n-2)/8}\,\mathrm{d}x =$$
$$\frac{-16}{\gamma^2(n-2)}e^{-x^2\gamma^2\pi(n-2)/8}\Big|_0^{\infty} = \frac{16}{\gamma^2(n-2)}\ .$$

The right term in (2) equals

$$\int_{x=0}^{\infty} g(x)\int_{y=0}^{\infty} g(y)2e^{-(x+y)^2\pi(n-2)/8}\,\mathrm{d}y\mathrm{d}x \le 2U^2\int_{x=0}^{\infty}\int_{y=0}^{\infty} e^{-(x+y)^2\pi(n-2)/8}\,\mathrm{d}y\mathrm{d}x =$$
$$2U^2\int_{z=0}^{\infty} ze^{-z^2\pi(n-2)/8}\,\mathrm{d}z = 2U^2\left(\frac{-4}{\pi(n-2)}e^{-z^2\pi(n-2)/8}\Big|_0^{\infty}\right) = \frac{8U^2}{\pi(n-2)}\ .$$

Lemma 7 *Let X be the random variable for the number of plausible pairs in a random set S of $n > 4$ points uniformly distributed in a convex region of area 1 and perimeter U. Then*

$$\mathrm{E}(X) < 4(n+2)(\frac{U^2}{\pi} + \frac{2}{\gamma^2}) = O(U^2 n)$$

and

$$\mathrm{E}(X\log X) \le 2\mathrm{E}(X)\log n = O(U^2 n\log n)\ .$$

The first bound follows from

$$\mathrm{E}(X) = \binom{n}{2}\mathrm{Prob}(\{p,q\} \text{ is plausible}) < \binom{n}{2}\frac{1}{n-2}\left(\frac{8U^2}{\pi}+\frac{16}{\gamma^2}\right) \le 4(n+2)\left(\frac{U^2}{\pi}+\frac{2}{\gamma^2}\right) .$$

For $\gamma = 1/\sqrt{5}$ and $U = 2\sqrt{\pi}$ (the perimeter of the disk of area 1) this gives $\mathrm{E}(X) < 56(n+2)$.

The second bound is now easily obtained by

$$\mathrm{E}(X\log X) = \sum_{m=1}^{\infty}(m\log m)\mathrm{Prob}(X=m) \le \sum_{m=1}^{\infty}(m\log\binom{n}{2})\mathrm{Prob}(X=m) < 2\mathrm{E}(X)\log n . \square$$

Let us call a pair $\{p,q\} \subset \mathcal{C}$ a *candidate* (for being plausible), if $r = r_{p,q} \le B$, where $B := \sqrt{(c\ln n)/(n-2)}$ for a constant c. Our next goal is now to show, that a random set does not contain too many candidates, and – with high probability – all plausible pairs are indeed candidates. We split again the problem by using the inequality

$$\mathrm{Prob}(r \le B) = \mathrm{Prob}(r_1 \le B \text{ or } r_2 \le B) < \mathrm{Prob}(r_1 \le B) + \mathrm{Prob}(r_2 \le B) .$$

We have

$$\mathrm{Prob}(r_1 \le B) = \mathrm{Prob}(|p-q| \le \frac{2B}{\gamma}) \le \left(\frac{2B}{\gamma}\right)^2 \pi = \frac{4\pi c\ln n}{\gamma^2(n-2)} ,$$

and

$$\mathrm{Prob}(r_2 \le B) = \mathrm{Prob}(\beta(p)+\beta(q) \le 2B) =$$
$$\int_{x=0}^{2B} g(x)\int_{y=0}^{2B-x} g(y)\,\mathrm{d}y\mathrm{d}x \le U^2\int_{x=0}^{2B}(2B-x)\,\mathrm{d}x = 2U^2B^2 = \frac{2U^2c\ln n}{n-2} .$$

Finally,

$$\mathrm{Prob}(\{p,q\} \text{ is plausible} \,|\, r > B) < 2e^{-B^2\pi(n-2)/2} = 2e^{-c\ln n\pi/2} = 2n^{-c\pi/2} . \quad (3)$$

Hence, the probability, that there is a plausible pair which is not a candidate is bounded by $\binom{n}{2}$ times the bound in (3). We summarize:

Lemma 8 *Let Y be the random variable for the number of candidates in a random set S of $n > 4$ points uniformly distributed in a convex region of area 1 and perimeter U. Then*

$$\mathrm{E}(Y) < \left(\frac{2\pi}{\gamma^2}+U^2\right)c(n+2)\ln n = O(U^2cn\log n) .$$

The probability that there exists a plausible pair that is not a candidate is bounded by

$$n^{2-c\pi/2} .$$

The bound for $\mathrm{E}(Y)$ follows from

$$\mathrm{E}(Y) = \binom{n}{2}\mathrm{Prob}(r \le B) < \binom{n}{2}\left(\frac{4\pi}{\gamma^2}+2U^2\right)\frac{c\ln n}{n-2} . \square$$

5 Greedy Triangulation Algorithms

These results lead to the following greedy triangulation algorithm:

5.1 Algorithm 1

Step 1: Generate all plausible pairs with $r_1 \leq B$. To do this, we generate all pairs of points separated by a distance of at most $2B/\gamma$. This is the fixed-radius-near-neighbors problem [2,7]. In this case a bucketing algorithm by Bentley, Stanat, and Williams can solve the problem in time $O(n + m)$, where m is the number of pairs that lie within $2B/\gamma$ of one another. As each pair is generated, test to see if it is plausible using the method described below.

Step 2: Generate all plausible pairs with $r_2 \leq B$. The easy way to do this is to find all points within $2B$ of the boundary of $\mathcal{C}$ and to generate all pairs. However, this can generate pairs with r_2 as long as $2B$. By sorting the points within $2B$ of the boundary of $\mathcal{C}$ in order of their distance from the boundary one can generate only the pairs needed by matching each point with points further from the boundary than itself only until $r_2 > B$, then going on to the next point. Also, note that pairs that also have $r_1 \leq B$ can be ignored, because they were generated in Step 1. Test each pair to see if it is plausible.

Step 3: Generate the greedy triangulation of the plausible pairs generated in Steps 1 and 2. To do this, first sort the pairs in increasing order of distance between the points. Start with an empty triangulation, and attempt to create an edge between each pair in turn, failing to create the edge if it fails the compatibility test described above.

5.2 Testing to See if a Pair is Plausible

We need a fast test to see if an edge is plausible. One way to do this uses a grid of squares. As a preprocessing step, cover $\mathcal{C}$ by a grid of $O(n)$ squares, each with side $1/\sqrt{n}$. For each bucket, create a list of the points in S that fall in that bucket. Then to test a pair (p, q), compute $D(p, q, r)$, with $r = \min(r_1, r_2)$ as described above. Go through the squares that overlap $D'(p, q, r)$ until you find a point lying in $D'(p, q, r)$ or find that no such point exists. Similarly go through the squares that overlap $D''(p, q, r)$ until you find a point lying in this half-disk or find that no such point exists. If either half-disk is empty, the point is plausible. In searching through the grid squares, start with the squares with maximum overlap with the half-disk.

For uniformly distributed points this test will run in $O(1)$ expected time. The probability that a grid square that lies completely in $\mathcal{C}$ contains no point from S is $(1 - 1/n)^n$. This is always less than e^{-1}. Therefore the probability that a given grid square contains a point is greater than $1 - e^{-1} > 0.63$. This implies that a search through grid squares looking for a non-empty one is expected to look at fewer than

2 squares. When r is small enough that no full grid squares overlap a half-disk, then only a constant number of grid squares overlap the half-disk. When r is larger, the number of grid squares examined before an point is found is expected be a constant.

Unfortunately, a bad distribution of points could cause this test to look at $O(n)$ squares for a very long edge. To prevent this, we stop testing after looking at $c \ln n$ squares for some c that we choose. Thus the test could occasionally decide that a half-disk is empty when it in fact is not. Thus a candidate could occasionally be considered plausible when it is not. But this will happen with probability less than $(1 - e^{-1})^{c \ln n} = n^{c(\ln(e-1)-1)}$. This is about $n^{-.46c}$. We can therefore choose c so that the expected number of edges on which the test fails is $o(1)$, and this limitation will avoid bad worst-case times without hurting expected times.

5.3 Analysis of Algorithm 1

Given this test for plausible pairs, it is easy to show that Algorithm 1 runs in $O(n \log n)$ expected time. By using the floor function we can do the preprocessing for the edge test in $O(n)$ time. By Lemma 8 the number of pairs considered in Steps 1 and 2 is $O(n \log n)$. Each test for plausibility takes constant expected time, so these steps require $O(n \log n)$ time. By Lemma 7 the number of pairs that are generated in Steps 1 and 2 that are plausible is $O(n)$. Sorting these takes $O(n \log n)$ time, and the compatibility test for each takes $O(1)$ time. Therefore we expect Step 3 and the entire algorithm require $O(n \log n)$ time and $O(n)$ space. In the worst case the algorithm could take $O(n \log^2 n)$ time and $O(n \log n)$ space.

Unfortunately, this algorithm is not guaranteed to generate the greedy triangulation. It will generate it with probability $1 - n^{2-c\pi/2}$, and because we can choose c as large as we like we can make this probability arbitrarily close to 1. But the algorithm could fail to produce a triangulation, either because neither diagonal of a quadrilateral was a candidate or because the edge compatibility test (which depends on the correctness of the partial triangulation) could fail if edges were missing. Alternately, it could produce a triangulation that was not the greedy one, and there is no known fast way to verify the correctness of a greedy triangulation.

Fortunately, a minor modification of the algorithm will eliminate this problem.

5.4 Algorithm 2

We turn Algorithm 1 into a two-phase algorithm. We run Steps 1 and 3 of Algorithm 1, but skip Step 2. What we will be left with at the end of this process is a partial greedy triangulation. All of the short edges will be present, but no edge longer than $2B/\gamma$ will be present. But Lemma 8 implies that with very high probability, all edges that have either endpoint at least $2B/\gamma$ from the boundary of $\mathcal{C}$ will be present. We will call points at least $2B/\gamma$ from the boundary *interior points*. We call a greedy triangulation edge with an interior point as (at least) one of its endpoints an *interior edge*. The expected number of interior edges that are missing is bounded by the expected number of plausible edges which are not candidates, so is at most

$n^{2-c\pi/2}$. As long as c is chosen to be at least $4/\pi$ this is $O(1)$ edges. This implies that at most $O(1)$ interior points will not be closed (as defined in the edge test section). This is because a missing edge can cause at most four points to be not closed (the four vertices of the quadrilateral with the missing edge as diagonal).

This observation leads to a new Step $2'$.

Step $2'$: Generate plausible long pairs Generate all possible pairs of non-closed points, and reject all implausible pairs. Sort these pairs, and continue running Step 3 of Algorithm 1 with these pairs as well.

Because all short pairs are tried and then all longer pairs that could possibly create an edge are tried, the algorithm will correctly generate the greedy triangulation of any set of points.

The edge test data structure keeps track of the list of incident open edges for each point. A point which is closed will have incident edges, but no open edges in its list.

Will this step generate too many candidate pairs? The number of interior points that are not closed is $O(1)$. Non-interior points lie within $2B/\gamma$ of the boundary of $\mathcal{C}$. The total number of non-interior points is expected to be at most $2BUn/\gamma$, because $2BU/\gamma$ is an upper bound on the area close enough to the boundary of $\mathcal{C}$. Therefore the total number of candidate edges generated in Step $2'$ is $\binom{2BUn/\gamma+O(1)}{2} = O(n \log n)$.

Lemma 7 says that the total expected number of plausible pairs is $O(n)$, so Step $2'$ is expected to generate $O(n)$ plausible pairs. Therefore sorting and testing these for compatability will take $O(n \log n)$ time.

This shows that Algorithm 2 will always compute the greedy triangulation and is expected to run in $O(n \log n)$ time, using $O(n)$ space. In the worst case it could take $O(n^2 \log n)$ time and $O(n^2)$ space.

5.5 Algorithm 3 – An Algorithm that Depends Less on the Uniform Distribution

Algorithm 2 depends heavily on the uniform distribution, both to get a fast expected-time test for plausibility and to tell when to end the first phase and begin the second. The following algorithm is a variant that is less sensitive to the exact distribution. It will dynamically decide when to switch from one phase to the next in an attempt to balance the amount of work done in each phase.

In the first phase it generates possible edges in increasing order using an algorithm of Dickerson, Drysdale, and Sack [8]. (Algorithms to enumerate the k closest interpoint pairs have been invented by Salowe and by Lenhof and Smid, but because they need to know k in advance they are less appropriate in this context [32, 17].) When the number of pairs of not closed points is proportional to the number of pairs already examined, it starts over, enumerating pairs of not closed points in increasing order (similar to Step $2'$ above).

How do we know when to switch over? We keep track of the number of points which are not closed. When during an edge insertion a point becomes closed, we subtract 1 from the number of non-closed points c. When $c^2/2$ is greater than the number of edges tested so far, the number of edges generated in the second phase will equal the number in the first, so we change to the second phase.

5.6 Analysis of Algorithm 3

Generating the next pair in increasing order requires $O(\log n)$ time, and a compatibility test takes $O(1)$ time. Therefore the algorithm runs in time $O(\log n)$ times the number of pairs generated. It requires space proportional to the number of pairs generated.

For the uniform distribution, we have already seen that all interior points are expected to be closed after examining $O(n \log n)$ edges, and that at that point the number of points which are not closed is $O(Bn) = O(\sqrt{n \log n})$. Therefore the number of pairs in the second phase will be $O(n \log n)$. This implies that the algorithm is expected to take $O(n \log^2 n)$ time and $O(n \log n)$ space for a uniform distribution.

6 Summary and Open Problems

We have given a new method for testing edge compatibility in a greedy triangulation. The method is based on Theorem 4, the CW/CCW chain theorem, which states an interesting property of greedy triangulations. Our method requires only $O(1)$ time for both the compatibility test and updates operations. This is a significant improvement over previous methods.

We then proved a necessary condition involving half-disks for an edge to be in the greedy triangulation. This lead to theorems on the number of pairs of points that were plausible and that were candidates to be plausible edges.

Finally, we used these characterizations and the compatibility test to prove the correctness and runtime of several new algorithms for computing the greedy triangulation. On uniformly distributed points we can compute the greedy triangulation in expected time $O(n \log n)$ and space $O(n)$.

Some obvious questions arise from this work.

Problem 1 *OPEN We can construct a point set for which Algorithm 3 would require $\Theta(n^2 \log n)$ time. However this set is highly structured and non-random. What is the expected run time for Algorithm 3 for random distributions other than the uniform distribution?*

Problem 2 *OPEN Is there a greedy triangulation algorithm requiring $o(n^2)$ time in the* worst *case? Note that with our edge test, a way to enumerate all pairs of points in increasing order by distance in $O(n^2)$ time would lead to an $O(n^2)$ algorithm for the GT, but would not lead to an $o(n^2)$ algorithm.*

Problem 3 *OPEN What is the true worst-case ratio for δ in Lemma 3? We have bounded it between $d(p,q)/(2\sqrt{5})$ and $d(p,q)/(2\sqrt{2})$.*

References

[1] A. Aggarwal and L. J. Guibas and J. Saxe and P. Shor, "A linear time algorithm for computing the Voronoi diagram of a convex polygon ." *Discrete Comput. Geom.* **4** (1989) 591–604.

[2] J. Bentley, D. Sanat and E. Williams Jr., "The complexity of finding fixed-radius near neighbors." *Information Processing Letters* **6** (1977) 209–213.

[3] R. C. Chang and R. C. T. Lee, "On the average length of Delaunay triangulations." *BIT* **24** (1984) 269–213.

[4] P. L. Chew, "Constrained Delaunay triangulations." *Proceedings of the Third Annual ACM Symposium on Computational Geometry* (1987) 215–222.

[5] Dehen, Flach, Sack, and Valiveti, "Analog Parallel Computational Geometry." *Proceedings of the 5th Canadian Conference on Computational Geometry* (1993) 143–153.

[6] M. Dickerson, "Expected rank of the longest edge in the greedy triangulation." *Proc. of the 4th Canadian Conference on Computational Geometry* (1992) 182–187.

[7] M. Dickerson and R. S. Drysdale, "Fixed-radius near neighbors search algorithms for points and segments." *Information Processing Letters* **35** (1990) 269–273.

[8] M. Dickerson, R.L. Drysdale, and J-R Sack, "Simple Algorithms for enumerating interpoint distances and finding k nearest neighbors." *International Journal of Computational Geometry and Applications* **3** (1992) 221–239.

[9] R. D. Düppe and H. J. Gottschalk, "Automatische Interpolation von Isolinien bei willküerlich verteilten Stützpunkten." *Allgemeine Vermessungsnachrichten* **77** (1970) 423–426.

[10] P. Gilbert, "New results in planar triangulations." MS Thesis, University of Illinois, Urbana, IL, 1979.

[11] S. Goldman, "A Space Efficient Greedy Triangulation Algorithm." *Information Processing Letters* **31** (1989) 191–196.

[12] L. Heath and S. Pemmaraju, "New results for the minimum weight triangulation problem." *Virginia Polytechnic Institute and State University, Department of Computer Science, TR 92-30* (1992). To appear in *Algorithmica.*

[13] R. Jarvis, "On the identification of the convex hull of a finite set of points in the plane." *Information Processing Letters* **2** (1973) 18–21.

[14] D. Kirkpatrick, "A note on Delaunay and optimal triangulations." *Information Processing Letters* **10** (1980) 127–128.

[15] G. Klincsek, "Minimal triangulations of polygonal domains." *Ann. Discrete Math.* **9** 121–123.

[16] D. T. Lee and A. K. Lin, "Generalized Delaunay triangulations for planar graphs." *Discrete Comput. Geom.* **1** (1986) 201–217.

[17] H. P. Lenhof, M. Smid, "Enumerating the k-closest pairs optimally." *Proceedings of the 33rd FOCS*, (1992) 380–386.

[18] C. Levcopoulos, "An $\Omega(\sqrt{n})$ Lower Bound for the Nonoptimality of the Greedy Triangulation." *Information Processing Letters* **25** (1987) 247–251.

[19] C. Levcopoulos and A. Lingas, "On Approximating Behavior of the Greedy Triangulation for Convex Polygons." *Algorithmica* **2** (1987) 175–193.

[20] C. Levcopoulos and A. Lingas, "Fast Algorithms for Greedy Triangulation." *BIT* **32** (1992) 280–296.

[21] C. Levcopoulos and A. Lingas, "Greedy Triangulation Approximates the Minimum Weight Triangulation and Can be Computed in Linear Time in the Average Case." Tech. Report LU-CS-TR:92-105, Dept. of Computer Science, Lund University, 1992. A preliminary version of this report appeared in *Proc. ICCI '91, LCNS 497.*

[22] A. Lingas, "The greedy and Delaunay triangulations are not bad in the average case." *Information Processing Letters* **22** (1986) 25–31.

[23] A. Lingas, "On Approximating Behavior and Implementation of the Greedy Triangulation for Convex Planar Point Sets." *Proceedings of the Second Annual ACM Symposium on Computational Geometry* (1986) 72–79.

[24] A. Lingas, "A new heuristic for the minimum weight triangulation." *SIAM Journal of Algebraic and Discrete Methods* **8** (1987) 646–658.

[25] A. Lingas, "Greedy triangulation can be efficiently implemented in the average case." *Proceedings of Graph-Theoretic Concepts in Computer Science* (1988) 253–261.

[26] E. Lloyd, "On triangulations of a set of points in the plane." *Proceedings of the 18th FOCS* (1977) 228–240.

[27] G. Manacher and A. Zobrist, "Neither the greedy nor the Delaunay triangulation of the planar set approximates the optimal triangulation." *IPL* **9** (1979) 31–34.

[28] G. Manacher and A. Zobrist, "Probabilistic methods with heaps for fast-average-case greedy algorithms." *Advances in Computing Research* vol. 1 (1983) 261–278.

[29] D. A. Plaisted and J. Hong, "A heuristic triangulation algorithm." *J. Algorithms* **8** (1987) 405–437.

[30] F. Preparata and M. Shamos, "Computational Geometry: an introduction." Springer-Verlag, 1985.

[31] A. Rényi and R. Sulanke, "Über die konvexe Hülle von n zufällig gewählten Punkten, I." *Z. Wahrsch Verw Gebiete* **2** (1963) 75–84.

[32] J.S. Salowe, "Enumerating distances in space." *Internat. J. Comput. Geometry Appl.* **2** (1992) 49–59

[33] C. A. Wang, "Efficiently updating the constrained Delaunay triangulations." *BIT* **33** (1993) 238–252.

[34] Y.F. Wang and J.K. Aggarwal, "Surface reconstruction and representation of 3-D scenes." *Pattern Recognition* **19** (1986) 197–207.

[35] C. A. Wang and L. Schubert, "An optimal algorithm for constructing the Delaunay triangulation of a set of line segment." *Proceedings of the Third Annual ACM Symposium on Computational Geometry* (1987) 223–232.

[36] P. Yoeli, "Compilation of data for computer-assisted relief cartography." In *Display and Analysis of Spatial Data* J.C.Davis and M.J. McCullagh, editors, John Wiley & Sons, NY (1975).