Fast greedy triangulation algorithms

Matthew T. Dickerson\textsuperscript{a,*,1}, Robert L. Scot Drysdale\textsuperscript{b,2}, Scott A. McElfresh\textsuperscript{b}, Emo Welzl\textsuperscript{c}

\textsuperscript{a} Department of Computer Science, Middlebury College, Middlebury, VT 05753, USA
\textsuperscript{b} Department of Mathematics and Computer Science, Dartmouth College, Hanover, NH 03755, USA
\textsuperscript{c} Institut für Informatik, Freie Universität, Berlin, Germany

Communicated by David Kirkpatrick; submitted 22 January 1994; accepted 21 April 1996

Abstract

We present a new method for testing compatibility of candidate edges in the greedy triangulation, and new results on the rank of edges in various triangulations. Our edge test requires $O(1)$ time for test and update, $O(n)$ space, and $O(n)$ time to initialize. Based on these results, we present fast greedy triangulation algorithms with expected case running time of $O(n \log n)$ for uniform distributions over convex regions. While algorithms with $O(n)$ expected case running times exist, the algorithms presented here are simpler to implement and work well in practice. © 1997 Elsevier Science B.V.

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 an algorithm that computes the greedy triangulation in expected time $O(n \log n)$ and space $O(n)$ for points uniformly distributed over any convex region. The algorithm is easy to implement and performs well in practice. 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. Furthermore, though some vertices can have $\Theta(n)$ adjacent edges, we are able to show that a list of at most 10 of these edges suffices for comparison of angular order. Our method therefore provides a $\Theta(1)$ time edge test, $\Theta(1)$ time update operation, $\Theta(n)$ time for initialization, and $\Theta(n)$ space. This compares favorably with the previous method of Gilbert [14], requiring $\Theta(\log n)$ 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 [33], but deterministically decides if a conflict exists rather than just finding a conflict with high probability.

We next prove that an edge can be greedy only if one of two small half-disks centered at its midpoint contains no points from $S$. This necessary condition for greedy edges allows us to prove a number of properties about the greedy triangulation for uniformly distributed points drawn from a convex compact region $C$. 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 $C$. 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 $C$ 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 expected $O(n \log n)$ 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 compatibility test.

This algorithm will run in $O(n \log n)$ expected time for uniformly distributed points in a convex region, and will produce the greedy triangulation with very high probability. 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 [26]. 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, but extending it to nonrectangular convex shapes would require modifications of the algorithm and analysis. Although their algorithm is beautiful theoretically, it has not been implemented and appears to be difficult to implement practically. Recent empirical results of [13], which uses Algorithm 2 described in this paper on sets of up to 10000 points, suggest that our algorithm is both easy to implement and practical for point sets of reasonable size.

Since the submission of this paper an extension of our ideas has been developed by Drysdale, Rote and Aichenholzer [11]. This modified algorithm runs in $\Theta(n)$ expected time. They use the same basic approach and the same edge test. By using a significantly more complicated method of generating candidate edges they are able to generate only $\Theta(n)$ of them rather than $\Theta(n \log n)$. They then use a version of radix sort to sort these edges by length in $\Theta(n)$ expected time.
1.2. Background

Efficiently computing the greedy triangulation is a problem of long standing, going back at least to 1970 [12]. A number of the properties of the GT have been discovered [24, 28, 32, 33] and the greedy algorithm has been used in applications [7, 33].

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 $\Theta(n^2)$ space and time,

$$T(n) = O(n^2 \log n + n^2 f(n) + ng(n)),$$

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 [35]. 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 [14] 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 [33] 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 [8]. 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) $\Theta(n^2)$ 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 initialization 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 [15] and by Lingas [30]. The method uses the generalized or constrained Delaunay triangulation [4, 20, 41]. 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 [30] 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 [25,38]. 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 [26]. These methods are elegant, but are significantly more complicated than our methods and should be slower for practical-sized problems.

Finally, Wang [39] and Levcopoulos and Krznaric [23] have recently developed algorithms that they claim will compute the greedy triangulation of general sets of points in $O(n \log n)$ time. These algorithms should resolve the asymptotic complexity of computing the greedy triangulation. However, they are quite complex, and the algorithms in this paper should prove easier to implement and faster for uniformly distributed point sets.

One use of the greedy triangulation is as an approximation to the minimum weight triangulation. 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 [28,31,35]. In a method suggested by Yoeli [42] for numerical approximation of bivariate data, the MWT provides a good approximation of the sought-after function surface. Also, both Wang and Aggarwal [40] and Barequet and Sharir [2] use an MWT in their algorithms to reconstruct surfaces from contours. In the method of [2], the MWT of a simple polygon is used only as a heuristic for producing a good reconstruction, and the $O(n^3)$ exact MWT algorithm is the bottleneck of an otherwise $O(n^2)$ time algorithm. Thus a fast and effective heuristic for the MWT would speed the overall algorithm significantly, and might also produce an equally good reconstruction. Though it has been shown how to compute the MWT in time $O(n^3)$ for the special case of $n$-vertex polygons [19], there are no known efficiently computable algorithms for the MWT in the general case [35]. We therefore seek efficiently computable approximations to the MWT.

A number of such heuristics are known, but those with the best proven bounds [34], as well as those which have been shown to work very well in practice [16,29] have impractical run times of $O(n^5)$ or $O(n^3)$. In fact, computing the greedy triangulation is a substep of the algorithm of [16]. A practical algorithm for the GT thus may be of interest as an approximation for the exact MWT. Neither the GT nor the Delaunay triangulation (DT) yields the MWT [18,31], but the GT is known to approximate the MWT to within a constant factor when the points lie on a convex polygon [24]. For points uniformly distributed in the unit square both the GT and the DT are expected to be within a constant factor of the MWT [5,26]. However, the GT seems to do a better job. In [13] the greedy triangulation was empirically shown to have lower weight than the DT and other fast approximations to the MWT.

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. For a points set $S$, we let $GT(S)$ be the greedy triangulation of $S$.

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 is not compatible with the partially constructed 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 2.1.** In an embedded 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 \( 1 \leq i < k \), \( p_i p_{i+1} \) 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 2.2.** 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 following observation will be used frequently in this paper.

**Observation 2.3.** 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 \( p_1 p_2 \) must have length \( \leq d(p_1, p_2) \).

This observations follows directly from the definition of the greedy algorithms which states that any edge compatible with previously added edges will be added to the triangulation. The new compatibility test method is based on the following lemma and theorem (see Fig. 1).

**Lemma 2.4.** Let \( S \) be a point set, \( \delta \) a distance, and \( T \) the set of all edges in \( \text{GT}(S) \) shorter than \( \delta \) union any set of edges in \( \text{GT}(S) \) of length \( \delta \). Let \( x, y, z \in S \) be points such that \( xyz \) forms a CW (respectively CCW) triangle and a CCW (respectively CW) chain in \( T \). Then the interior of triangle \( xyz \) contains no points from \( S \). If \( d(x, z) < \delta \) then \( xz \in T \). If \( d(x, z) = \delta \), then either \( xz \in T \) or \( xz \) is compatible with \( T \).

**Proof.** We first show by contradiction that there are no points in the interior of triangle \( xyz \). Assume there is at least one point interior to triangle \( xyz \), and let \( w \) be that interior point farthest from \( xz \). Let \( l \) be the line parallel to \( xz \) and passing through \( w \). Since \( xyz \) is a CCW chain in \( T \), we know that \( yw \) is not in \( T \). But \( yw \) is shorter than \( \max(yx, yz) \) and thus shorter than \( \delta \), and therefore \( yw \) is not in \( \text{GT}(S) \) either. So by Observation 2.3, some edge shorter than \( d(y, w) \) must intersect \( yw \). But there is no edge crossing \( xy \) or \( yz \) and by our assumption no point interior to \( xyz \) lies on the same side of \( l \) as \( y \), and thus no such edge can exist which crosses \( yw \). This contradiction shows that no points lie interior to triangle \( xyz \). Since no edge can cross \( xy \) or \( yz \) and no edge can have an endpoint interior to triangle \( xyz \), it follows that any edge in \( T \) intersecting \( xz \) must have \( y \) as an endpoint. But \( y \) cannot be an endpoint of such an edge because by our assumption \( xyz \) is a CCW chain in \( T \). Therefore no

![Fig. 1. CCW chain with respect to (and intersecting) segment \( pq \).](image-url)
edge in $T$ properly intersects $xz$ and $xz$ is compatible with $T$ if it is not already in $T$. If $d(x, z) < \delta$ then $xz \in T$ by Observation 2.3.

**Theorem 2.5** (Clockwise/Counter-clockwise Chain Theorem). Given a set of points $S$ and a distance $\delta$, let $p, q \in S$ be a pair of points with $d(p, q) = \delta$, and let $T$ be the set of all edges in $GT(S)$ shorter than $\delta$ union any set of edges in $GT(S) - \{pq\}$ of length $\delta$. Then $pq$ is compatible with $T$ if and only if one of the following two conditions holds:

1. $p$ and $q$ are endpoints of a two edge CW chain in $T$; or
2. $T$ contains no CW or CCW chain that has $p$ or $q$ as an endpoint and a second edge of the chain intersecting $pq$, and $T$ contains no CW or CCW chain with $p$ (respectively $q$) as an endpoint and the next point on the chain closer than $\delta$ to $q$ (respectively $p$).

**Proof.** Assume that $pq$ is not in $GT(S)$. We first show the existence of a CW or CCW chain $p, v_1, \ldots, v_k, x$ (respectively $q, v_1, \ldots, v_k, x$) that has the following five properties.

1. edge $v_kx$ 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 $v_kx$) form a convex chain; and
5. $v_kx$ is the closest edge to $p$ (respectively $q$) of all edges in $T$ intersecting $pq$.

(i). If $pq$ is not compatible with $T$, then there is an edge in $T$ intersecting $pq$. Without loss of generality, let this edge intersect $pq$ 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, also rearrange the plane so that $pq$ is the horizontal axis with $p$ on the left. Let $ab$ be the edge in $T$ 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 we let $v_k = a$ and $x = b$.
2. $a$ falls inside of $R$ but $d(p, a) \geq \delta$. We let $v_k = b$ and $x = a$. Note that $d(q, x) > \delta$, since $x$ is at least as close to $p$ as to $q$. But $d(x, v_k) < \delta$, and furthermore $xv_k$ intersects $pq$. This is possible only if $d(p, v_k) < \delta$.
3. Finally, $a$ could fall outside of $R$. Again we let $v_k = b$ and $x = a$. Because $xv_k$ intersects $pq$, $x$ must lie strictly to the left of $R$. (If it were above or below $R$ it would be too short to cross $pq$.) But then $v_k$ must lie in $R$, and the triangle inequality allows us to show that $d(p, v_k) < \delta$.

Note that for all three cases, properties (1) and (5) above have been met, and property (2) holds for $v_k$.

(ii). We have now described an edge $v_kx$ satisfying properties (1) and (5) above. We show that there is CCW chain from $p$ to $x$ containing edge $v_kx$ and meeting properties (2) through (4). Label the point where $xv_k$ and $pq$ intersect as $o$. Since $d(p, v_k) \leq \delta$ and $d(p, o) \leq \delta/2$, for every point $y$ in triangle $pou_k$ (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 $ouv_k$ is a segment of this edge, and therefore there are no edges in $T$ intersecting $po$ or $ou_k$. We now show how to construct the CCW chain in a fashion similar to the “gift-wrapping” approach of the Jarvis march [17]. Let $v_1 \in S$ be the point in triangle $\triangle pou_k$ that minimizes the counterclockwise angle of $pv_1$ with respect to $pq$. (If $\triangle pou_k$ 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 $\triangle pou_k$ below ray $pv_1$, and there are no edges intersecting $po$ or $ou_k$. 
It follows that \(pv_1\) is in \(T\), 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 satisfies property (1) of the theorem. Furthermore, every point along this chain falls in \(\triangle pov_k\) and therefore satisfies property (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 property (4) is satisfied 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 \(p, v_1, \ldots, v_k, x\) satisfies property (3) as well. Assume that \(k > 1\) and that point \(v_1\) does not lie strictly inside the circle \(C\) of radius \(\delta\) centered at \(q\). (See Fig. 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 away from \(pq\), 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 2.4 that \(v_{k-1}x\) must be an edge in \(T\). 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 $x$.

We have shown that if $pq$ is not in $T$ then there is a CW or CCW chain with the five given properties. It remains now to show how our theorem follows from this observation. We prove first the “if” clause. Assume that one of the two conditions of our theorem holds. First assume condition (1) that there is a CW chain in $T$ of two edges connecting $p$ and $q$. In this case we know directly from Lemma 2.4 that $pq$ is compatible with $T$. So instead assume that condition (1) fails but condition (2) holds: if any CW or CCW chain exists in $T$ having an endpoint on $p$ (respectively $q$) then the second edge of the chain does not intersect $pq$ and the vertex $v_1$ adjacent to $p$ (respectively $q$) is a distance at least $\delta$ from $q$ (respectively $p$). But if for every CW or CCW chain from $p$ we have $d(v_1,q) \geq \delta$ and the second edge of the chain does not intersect $pq$, then property (3) fails and the edge $pq$ must be compatible with $T$.

We conclude with the “only if” part of the proof. Assume that $pq$ is compatible with $T$. Let $v_1$ and $v_2$ be the first two vertices of the CCW chain in $T$ with respect to $qp$. If $v_2 = q$, then we have satisfied condition (1) and we are done. If no chain at all exists, then we have satisfied condition (2) for the CCW chain from $p$, and we proceed to the three symmetric cases. If no CW chain or CCW chain exists from $p$ or $q$, then we have satisfied condition (2). So assume there is a CCW chain $p, v_1, v_2$ with $v_2 \neq q$. We know that no edge (including the second) intersects $pq$, so if $d(q,v_1) \geq \delta$ then we have also satisfied condition (2). So let $d(q,v_1) < \delta$ and $v_2 \neq q$. We may continue to assume that no edge intersects $pq$. Since $p, v_1$ is a CCW chain in $T$ with respect to $qp$, it follows that $q, p, v_1$ is a CCW chain in $T$. So by Lemma 2.4, the triangle $qpv_1$ could contain no points in its interior and $v_1q$ must be compatible with $T$ also. Since $d(v_1,q) \leq \delta$, we know that it is also in $T$. But then in $T$ we would have that $v_1q$ is the next edge CCW around $v_1$ from $pv_1$, contradicting the assumption that $v_2 \neq q$. \[ \square \]

The following is a direct corollary of the previous theorem.

**Theorem 2.6 (Corollary).** If $pq$ is not compatible with $T$, then $T$ contains a CW or CCW chain $C$ meeting one of the following conditions:

1. $p$ or $q$ is an endpoint of $C$ and the second edge of $C$ properly intersects $pq$; or
2. $p$ (respectively $q$) is an endpoint of $C$ and the next point on the chain is closer than $\delta$ to $q$ (respectively $p$).

**2.1. Edge test method and proof of correctness**

We now give a fast greedy triangulation edge test method based on Theorem 2.5. 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 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 $v_2$. **
(lb) IF angle $qpv_1 \geq \pi/2$ OR no CCW chain exists, THEN goto Step 2.
(lc) IF $v_1v_2$ intersects $pq$ THEN return FALSE ($pq$ is incompatible).
(ld) IF $v_2 = q$ THEN return TRUE (pair $(p, q)$ is compatible with $T$).
(le) 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.

The remaining Steps 2–4 are symmetric, except if the condition in (4e) is false we return TRUE instead of jumping to a new step. The proof of correctness follows directly from Theorem 2.5.

2.2. Implementation and analysis

We now discuss the implementation and efficiency of our edge test. We store our greedy triangulation $T$ using adjacency lists. Each adjacency list is represented as a circular linked list of edges 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 with respect to each endpoint $p$. If $n_p$ is the number of edges adjacent to $p$ in $T$, then this would normally take $\Theta(n_p)$ time, or $\Theta(\log n_p)$ 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 $\alpha$ 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 2.4 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.

The important observation is that 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 $\alpha$ is not the largest angle in triangle $pxy$, we know that $xy$ is not the longest edge in the triangle. By Lemma 2.4, $xy$ is in $T$, which means that $(x, y)$ is closed. Therefore every open pair has $\alpha \geq \pi/3$, 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 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 $2\pi$. 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 which is \(\Theta(1)\). 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, recall that the method of Gilbert [14] requires \(O(n^2 \log n)\) time for initialization, \(O(n)\) for an edge test, \(O(n \log n)\) time for an update, and \(O(n^2)\) space.

3. A necessary condition for greedy triangulation edges

We begin by stating a simple and obvious lemma.

**Lemma 3.1 (Convex Quadrilateral Rule).** Let \(pqr\) and \(pqrs\) 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.** This property is also known as "local optimality" [16]. The proof follows directly from Observation 2.3. We now give an important (though less obvious) lemma\(^3\) that states a necessary (but not sufficient) condition for an edge to be a greedy triangulation edge. This lemma is similar to [26, Lemma 3.1] and to [6, Theorem 3].

**Lemma 3.2.** 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 Fig. 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 \(D\). (Note that \(d(a, b) \leq 2\delta\).) Let \(C\) be the circle with \(pq\) as a diameter, and let \(\alpha\) be the length of the shortest segment with endpoints on or outside of \(C\) that passes through \(D\). Then \(\alpha = \sqrt{d(p, q)^2 - 4\delta^2} = 4\delta\).

\(^3\) An earlier false version of this lemma with \(\delta = d(p, q)/2\) was given in [8]. We give the corrected version here.
We will assume that \( pq \) is a GT edge and show that this leads to a contradiction. If \( pq \) is a GT edge, then \( ab \) cannot be a GT edge because they would intersect. Therefore, by Observation 2.3 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 \( pq \).

Since there are points on both sides of edge \( pq \), it is not a convex hull edge, and therefore it must be an edge in two triangles – an upper and a lower. Let \( p\bar{w}q \) be the upper triangle and let \( pz\bar{l}q \) be the lower triangle.

If \( \bar{w}l \) and \( \bar{z}l \) are both in \( C \), then \( d(\bar{w}l, \bar{z}l) < d(p, q) \) and thus by Lemma 3.1 \( pq \) 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 \( p\bar{w} \) 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 \( \overrightarrow{ba} \). 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 \( \overrightarrow{ab} \) 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 then all of them would be longer than \( \alpha \). But they comprise all of the edges crossing \( ab \), so by Observation 2.3 one of them must be shorter than \( d(a, b) < 2\delta \). This contradicts the fact that \( \alpha = 4\delta \). 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 3.3.** 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 \( \alpha \). Let \( f_i \) be the lowest-numbered \( f_j \) with an endpoint in \( C \), and let \( y \) be an endpoint of \( f_i \) in \( C \). Then all edges crossing \( ya \) are at least \( \alpha \) long (because neither endpoint lies in the interior of \( C \)), but \( d(y, a) \)
must be shorter than α because y is in C and a is in D. (The maximum distance that y could be from a is \(d(p, q)(1 + 1/\sqrt{5})/2\) or about .72\(d(p, q)\) while α is about .89\(d(p, q)\).) This contradicts Observation 2.3. A symmetric argument shows that one of the \(f_j\) edges must also have an endpoint in C.

**Observation 3.4.** If \(e_k\) has an endpoint in C and \(e_1, \ldots, e_{k-1}\) have both endpoints outside of C, then \(e_1, \ldots, e_{k-1}\) 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 \(D\), in the sense that \(xw_{k-1}\) intersects \(D\) and \(x\) lies outside of \(D\) on the part of the segment to the left of this region of intersection. Finally, for some \(1 \leq l \leq k - 1\) there is a point \(w_l\) that lies within \(d(p, q)\) of \(p\). (See Fig. 3 again.)

This observation follows from several points. First, no \(w_i\) can lie to the left of ray \(\overline{qw_l}\) 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 \(\overline{qw_l}\) 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 3.3. 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 \(e_1, \ldots, e_{k-1}\).

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 2.3 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 \(p\) tangent to \(D\) from above.

Observation 3.3 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 \(\alpha\). If it is in the right half of \(C\), its distance to \(w_l\) will be less than \(\alpha\). In either case we have a contradiction, because all intervening edges will be at least \(\alpha\) long. (All of them have both endpoints on or outside of \(C\).)

**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 the midpoint of segment \(pq\). 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 $0 < \gamma \leq 1$, 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 $r_1 = \gamma d(p, q)/2$. The previous section showed that if $\gamma = 1/\sqrt{5}$, only plausible 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, uniformly distributed in a convex compact region $C$. We normalize $C$ to be of area 1 to simplify the notation.

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

$$\Pr(D'(p, q, r_1) \text{ is empty or } D''(p, q, r_1) \text{ is empty}) = 2 \left(1 - \frac{r_1^2 \pi}{2}\right)^{n-2} < 2e^{-r_1^2 \pi (n-2)/2}.$$  

We also have to cope with pairs of points $\{p, q\}$ where $D(p, q, r_1)$ extends beyond the boundaries of $C$. For $p \in C$, let $\beta(p)$ denote the distance of $p$ from the boundary of $C$, or, in other words, the largest radius of a disk centered at $p$ which is still contained in $C$. What we would like to know is the largest radius of a disk centered at the midpoint of $pq$ which is still contained in $C$. The exact value of this radius depends on the shape of $C$ and the location of $p$ and $q$, but we can use convexity to get a lower bound for it. We observe that for any pair of points $p$ and $q$ in $C$, the disk $D(p, q, r_2)$, $r_2 = (\beta(p) + \beta(q))/2$, is contained in $C$. (This is the only use we make of the convexity of $C$, but without it or some similar assumption we could not derive the bounds in this section. Nonconvex regions can have $\Theta(n^2)$ expected plausible pairs.) It follows that for any pair $\{p, q\} \subset C$ the probability of being plausible (after adding $(n-2)$ random points) is bounded by

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

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 $C$. Then the probability of being plausible is bounded by

$$\int_0^\infty \int_{C} 2e^{-r^2 \pi (n-2)/2} dq dp < \int_0^\infty \int_{C} 2e^{-r_1^2 \pi (n-2)/2} dq dp + \int_0^\infty \int_{C} 2e^{-r_2^2 \pi (n-2)/2} dq dp. \quad (2)$$

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

Now the left term in (2) equals

$$\int_0^\infty \int_{C} 2f_p(x)e^{-x^2 \gamma^2 \pi (n-2)/8} dx dp \leq \int_0^\infty 4x \pi e^{-x^2 \gamma^2 \pi (n-2)/8} dx = \frac{-16}{\gamma^2 (n-2)} e^{-\gamma^2 \pi (n-2)/8} \bigg|_0^\infty = \frac{16}{\gamma^2 (n-2)}.$$
The right term in (2) equals
\[
\int_0^\infty \int_0^\infty g(x)g(y)e^{-(x+y)^2\pi(n-2)/8} \, dy \, dx \leq 2U^2 \int_0^\infty \int_0^\infty e^{-(x+y)^2\pi(n-2)/8} \, dy \, dx \\
= 2U^2 \int_0^\infty z e^{-z^2\pi(n-2)/8} \, dz \\
= 2U^2 \left( \frac{-4}{\pi(n-2)} e^{-\pi(n-2)/8} \right) = \frac{8U^2}{\pi(n-2)}.
\]

**Lemma 4.1.** 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
\[
E(X) < 4(n+2) \left( \frac{U^2}{\pi} + \frac{2}{\gamma^2} \right) = O(U^2n) \quad \text{and} \quad E(X \log X) \leq 2E(X) \log n = O(U^2n \log n).
\]

**Proof.** The first bound follows from
\[
E(X) = \binom{n}{2} \Pr(\{p,q\} \text{ is plausible}) < \binom{n}{2} \frac{1}{n-2} \left( \frac{8U^2}{\pi} + \frac{16}{\gamma^2} \right) \leq 4(n+2) \left( \frac{U^2}{\pi} + \frac{2}{\gamma^2} \right).
\]
For \( \gamma = 1/\sqrt{5} \) and \( U = 2\sqrt{\pi} \) (the perimeter of the disk of area 1) this gives \( E(X) < 56(n+2) \).

The second bound is now easily obtained by
\[
E(X \log X) = \sum_{m=1}^\infty (m \log m) \Pr(X = m) \leq \sum_{m=1}^\infty \left( m \log \binom{n}{2} \right) \Pr(X = m) < 2E(X) \log n.
\]

Let us call a pair \( \{p,q\} \subset C \) a candidate (for being plausible), if \( r = r_{p,q} \leq B \), where \( B := \sqrt{\ln n/(n-2)} \) for a constant \( c \). Our next goal is 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
\[
\Pr(r \leq B) = \Pr(r_1 \leq B \text{ or } r_2 \leq B) < \Pr(r_1 \leq B) + \Pr(r_2 \leq B).
\]
We have
\[
\Pr(r_1 \leq B) = \Pr\left( |p - q| \leq \frac{2B}{\gamma} \right) \leq \left( \frac{2B}{\gamma} \right)^2 = \frac{4\pi \ln n}{\gamma^2(n-2)}
\]
and
\[
\Pr(r_2 \leq B) = \Pr(\beta(p) + \beta(q) \leq 2B)
\]
\[
= \int_0^{2B} \int_0^{2B-x} g(x)g(y) \, dy \, dx \leq U^2 \int_0^{2B} (2B - x) \, dx = 2U^2 B^2 = \frac{2U^2 \ln n}{n-2}.
\]
Finally,

\[ \Pr(\{p, q\} \text{ is plausible } | \ r > B) < 2e^{-B^2\pi(n-2)/2} = 2e^{-c\ln n\pi/2} = 2n^{-c\pi/2}. \]  

(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 with the following result.

**Lemma 4.2.** 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

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

(where \( c \) and \( \gamma \) are constants involved in the definition of “candidate” and “plausible”).

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

\[ \frac{n^2-c\pi/2}{n^2-c\pi/2}. \]

The bound for \( E(Y) \) follows from

\[ E(Y) = \binom{n}{2} \Pr(r \leq B) < \binom{n}{2} \left( \frac{4\pi}{\gamma^2} + 2U^2 \right) \frac{c\ln n}{n-2}. \]

\[ \square \]

5. **Greedy triangulation algorithms**

These results lead to the following greedy triangulation algorithm:

5.1. **Algorithm 1**

**Step 1.** Generate all plausible pairs with \( r_1 < B \). 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 [3,9]. 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 \( r_2 < B \). The easy way to do this is to find all points within \( 2B \) of the boundary of \( C \) and to generate all pairs. However, this can generate pairs with \( r_2 \) as large as \( 2B \). By sorting the points within \( 2B \) of the boundary of \( C \) 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.

This description assumes that we know \( C \) and can find the distance of a point in \( S \) to the closest point on the boundary of \( C \) quickly (that is, in total time \( O(n \log n) \) for all \( n \) points). If this is not the case, an alternative is to compute the convex hull of \( S \) and to use this convex hull in place of \( C \).\footnote{This idea was suggested by a member of the audience at a talk given at the Max Plank Institute in Saarbrücken.} Computing the convex hull and then using bucketing to find points near its boundary would
work well, and would be practical to implement. Details of how this can be done are described in [11].

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 $C$ 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 $C$ 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 examined before an point is found is expected to be a constant.

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 4.2 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 4.1 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 Step 3 and the entire algorithm require $O(n \log n)$ time and $O(n)$ space.

Unfortunately, this algorithm is not guaranteed to generate the greedy triangulation. It will generate it with probability $1 - n^{2-cn/2}$, 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 easy way to verify the correctness of a purported 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 4.2 implies that with very high probability, all edges that have either endpoint at least $2B/\gamma$ from the boundary of $C$ 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 $n^2 - c_\pi^2/2$. 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 nonclosed 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 $C$. The total number of non-interior points is expected to be at most $2BU$/2, because $2BU/\gamma$ is an upper bound on the area close enough to the boundary of $C$. Therefore the total number of candidate edges generated in Step 2' is

$$\binom{2BU/n/\gamma + O(1)}{2} = O(n \log n).$$

Lemma 4.1 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 compatibility 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.

Empirical studies reported in [13] show that a good value for $c$ is between 0.005 and 0.1, which is much smaller than our analysis would suggest. It appears that most of the greedy edges are short and that it is faster to go to the second phase sooner and do more work there than to balance things out. These studies also suggest that for up to 25,000 points that using the plausibility test to eliminate candidate edges can be slower than sorting all candidates and testing them for compatibility.

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 [10]. (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 [21,37].) 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 nonclosed 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 edges 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 2.5, 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 6.1** (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 nonrandom. What is the expected run time for Algorithm 3 for random distributions other than the uniform distribution?

**Problem 6.2** (Open). What is the true worst-case ratio for \(\delta\) in Lemma 3? We have bounded it between \(d(p, q)/(2\sqrt{5})\) and \(d(p, q)/(2\sqrt{2})\).
References