Eurographics Symposium on Geometry Processing (2007), pp. 1–10
Alexander Belyaev, Michael Garland (Editors)

Triangulations with Locally Optimal Steiner Points

Hale Erten Alper Üngör

Dept. of Computer & Information Science & Engineering, University of Florida, Gainesville, FL, U.S.A.
{herten,ungor}@cise.ufl.edu

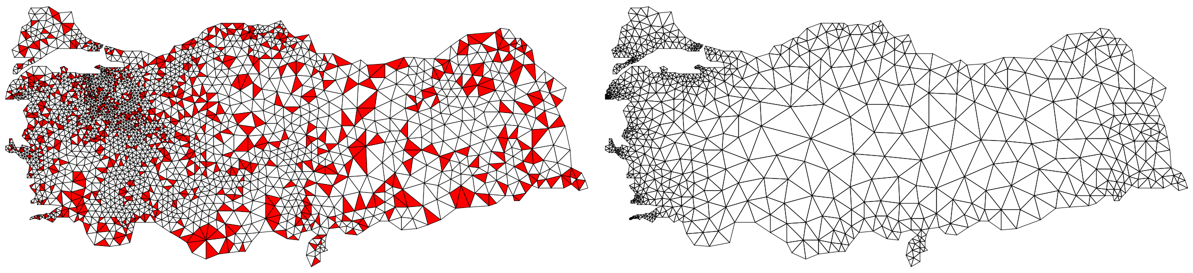

Figure 1: *The original Delaunay refinement algorithm chokes with bad triangles (shaded/red) for constraint angles larger than* 34°. *Our algorithm computes good triangulations (right) with constraint angles up to* 42°.

Abstract

We present two new Delaunay refinement algorithms, second an extension of the first. For a given input domain (a set of points or a planar straight line graph), and a threshold angle α*, the Delaunay refinement algorithms compute triangulations that have all angles at least* α*. Our algorithms have the same theoretical guarantees as the previous Delaunay refinement algorithms. The original Delaunay refinement algorithm of Ruppert is proven to terminate with size-optimal quality triangulations for* $\alpha \leq 20.7^\circ$*. In practice, it generally works for* $\alpha \leq 34^\circ$ *and fails to terminate for larger constraint angles. The new variant of the Delaunay refinement algorithm generally terminates for constraint angles up to* 42°*. Experiments also indicate that our algorithm computes significantly (almost by a factor of two) smaller triangulations than the output of the previous Delaunay refinement algorithms.*

Categories and Subject Descriptors (according to ACM CCS): F.2.2 [Nonnumerical Algorithms and Problems]: Geometrical problems and computations I.3.5 [Computational Geometry and Object Modeling]: Geometric algorithms, languages, and systems

1. INTRODUCTION

We revisit the following well-known two-dimensional geometric optimization problem: *Compute the smallest size triangulation of a given two dimensional domain (collection of points and/or segments) such that all the triangles in the triangulation are of good quality.* Naturally, we are allowed to introduce points (called the *Steiner points*) additional to the input points. This problem is also known as the simplicial mesh generation problem, or the quality Steiner triangulation problem [BE92, Ede01, She02, TW00]. Quality constraint of the problem is motivated by the numerical methods used in engineering applications where these triangulations are heavily used. A triangle is said to be good if its smallest angle is bounded from below. Small size objective is crucial for the applications as well, for obvious efficiency reasons.

Several (approximation) algorithms have been suggested for this problem. Quadtree refinement algorithms [BEG94] and Delaunay refinement algorithms [Che89, Rup93, Üng04] provide similar theoretical guarantees. In general for a specific algorithm, these guarantees can be stated as: *Given a two dimensional domain and a threshold angle* $\alpha \leq \gamma$*, the algorithm computes a triangulation of the input domain whose* *size is within a constant factor of the optimal such that all the angles of the triangulation are at least* α. The angle γ should be thought of as the theoretical limit of an algorithm. The exact value of the γ depends on the approach/algorithm as well as the input type. For instance, the Delaunay refinement algorithm of Ruppert [Rup93] is proven to compute good triangulations for $\alpha < \gamma = 30^\circ$ on point sets and for $\alpha < \gamma = 20.7^\circ$ on planar straight line graphs.

Lack of theoretical guarantee for a mesh refinement algorithm, when $\alpha > \gamma$, shows in practice as a never-ending refinement process. Steiner points are iteratively inserted until the computer reaches out of its numerical precision capacity or its memory. Shewchuk's experimental study revealed that in practice Delaunay refinement algorithm works better than its theoretical guarantee [She96]:

> "Ruppert [Rup93] proves that this procedure halts for angle constraint of up to 20.7°. In practice, the algorithm generally halts with an angle constraint of 33.8°, but often fails to terminate given an angle constraint of 33.9°. It would be interesting to discover why the cutoff falls there."

There has been attempts to explain this cutoff [MPW05]. Here, we take a different strategy and rather than explaining this limitation, we remove it. We present a new Delaunay refinement algorithm which sets a new cut-off. Our refinement algorithm terminates with good triangulations for constraint angles up to 42°.

1.1. Previous Work

1.1.1. Delaunay refinement

Delaunay refinement method involves first computing an initial Delaunay triangulation of the input domain, and then iteratively adding points called *Steiner points* to improve the quality of the triangulation. Traditionally, circumcenters of bad triangles are used as Steiner points [Che89, Rup93].

Circumcenters. Circumcenters of bad triangles, as studied by many [Che89, Rup93, She02], is a natural choice for improving the quality of a (Delaunay) triangulation through iterative refinement. Insertion of the circumcenter of a bad triangle, surely removes that bad triangle (thanks to empty circle property of Delaunay triangulations.)

Sinks. Edelsbrunner and Guoy [EG01] suggested using *sinks*, circumcenters of acute triangles. For each triangle, an iterative walk in the triangulation each time crossing the edge opposite to the unique obtuse angle leads to a sink. Sinks of bad triangles are used as Steiner points. Note that sinks are at the local maxima of the local feature size function. Also note that sink of a bad triangle could be quite far away from the triangle. Hence, a bad triangle may remain bad after the insertion of its sink.

Offcenters. Üngör recently introduced a new type of Steiner points, called *off-centers*, as an alternative to circumcenters [Üng04]. Offcenters and circumcenters, however, differ from each other only for "very" bad triangles (those with smallest angle at most $\alpha/2$, and are the same for almost good triangles (those with the smallest angle between $\alpha/2$ and α. In practice, off-center insertion algorithm results in significant reduction in the output size [Üng04]. Off-centers are also numerically more stable than circumcenters and facilitates more robust software. The idea of using offcenters also leads to the design of the first time-optimal Delaunay refinement algorithm [HPÜ05].

Delaunay refinement software. The popular mesh generation software `Triangle`[†] is a robust implementation of the Delaunay refinement method. The software used Ruppert's circumcenter insertion algorithm [Rup93] in it first four releases and has been using Üngör's offcenter insertion algorithm [Üng04] in its latest two releases.

1.1.2. Other mesh refinement algorithms

Quadtree refinement. Quadtree methods provide similar guarantees as Delaunay refinement algorithms [BEG94]. However, in practice quadtree refined meshes are significantly larger than Delaunay meshes. Moreover, quadtree implementations lack the parameterization of the threshold angle.

Longest-edge propagation path (LEPP). There are other refinement strategies which are used in conjunction with Delaunay triangulations. For instance, the longest-edge propagation path (LEPP) methods use the midpoint of the longest edge of certain triangles as the Steiner point [RH99]. Theoretical bounds for these methods tend to be relatively weaker than that of the Delaunay refinement methods discussed above. Moreover, these methods also suffer from the termination problem in practice even for smaller angle thresholds than 34°.

1.2. Our idea and contribution.

All existing mesh refinement algorithms, including the circumcenter and the offcenter insertion variants of the Delaunay refinement, suffer from a so called *termination problem* for large values of α. In this paper, we explore ideas to alleviate this problem.

We propose two simple algorithms, second is an extension of the first. The first algorithm suggests to locate Steiner points inside a particular neighborhood of bad triangles that are furthest away from existing vertices. One might wonder "Are such points not circumcenters (at least for the almost good triangles)?". It turns out the answer is "No!", for most of the bad triangles in a typical Delaunay triangulation. In Section 3, we classify this Steiner point description into four different types.

[†] `http://www-2.cs.cmu.edu/~quake/triangle.html`

Our second algorithm incorporates a simple relocation scheme which becomes effective especially for large α values. Before attempting to insert a Steiner point for a bad triangle, we simply try fixing the bad triangle by relocating one of its vertices (if that vertex was inserted as a Steiner point).

These two simple ideas lead to the following as the main contributions of this paper:

- We give a definition of Steiner points which unifies the aforementioned Steiner point insertion strategies. The new rule automatically selects one of the four types of Steiner points, three of which (circumcenters, sinks, offcenters) are previously studied.
- With the new Steiner point definition, we extend the practical angle bound of the Delaunay refinement algorithms. While the previous Delaunay refinement algorithms fail to compute meshes, for threshold angles as small as 33.9°, the new algorithm computes good triangulations for threshold angles as high as 42°. On average we extended the practical angle threshold bound by 7°.
- The key idea in our Steiner point selection method is to stay away from the existing vertices. It turns out that circumcenters are not the best choice for this objective. Indeed, experiments give a surprising low percentage use for circumcenters among the four alternatives. We pick locations that result in longer edge lengths for the triangles of the mesh. As a result, the size of the output of our method is significantly smaller than that of the previous Delaunay refinement algorithms.
- We integrated a simple Steiner point relocation strategy to the refinement algorithm.

2. MOTIVATION

We explore ways to improve the practical performance of Delaunay refinement method. Existing versions of Delaunay refinement method becomes impractical when meshes with large smallest angles are desired. Two reasons behind this are explained below.

2.1. Termination Problem

Termination guarantee of the Delaunay refinement algorithms relies on a packing argument. This argument applies for small values of α, (i.e. $\alpha \leq 30^\circ$) where circumcenter insertion does not gradually decrease the shortest pairwise vertex distance. For $\alpha > 30^\circ$, however, the iterative refinement process could introduce smaller and smaller features and may not terminate (See Figure 2.) In practice, such phenomenon starts occurring for $\alpha > 34^\circ$, regardless of the type of input data. See Figure 3 and Figure 4.

2.1.1. Different Versions.

As described in Figure 2 any refinement algorithm that makes significant use of circumcenters is expected to suffer from the termination problem for threshold angles not much larger than $\alpha > 30^\circ$. Majority of the points inserted by most Delaunay refinement algorithms are circumcenters [Che89, EG01, Rup93, She02]. While the circumcenters are less frequently used by the offcenter insertion algorithm [Üng04], their percentage is significant enough that the practical performance bound remain roughly the same. Figure 3 illustrates how both `Triangle 1.4` which is an implementation of the circumcenter insertion, and `Triangle 1.6` which is an implementation of the offcenter insertion, suffer from the termination problem. It is important to observe the characteristic difference between these two methods as they get caught in the never-ending refinement process. This difference is due to order the bad triangles are handled. The offcenter insertion algorithm works better the bad triangles with the shortest edges are handled first. [HPÜ05]. On the other hand, the circumcenter insertion algorithm works better when the bad triangles with the smallest angles are handled first. When they terminate the offcenter insertion performs better than the circumcenter insertion, that is, it outputs meshes with fewer triangles. However, when interrupted in the never-ending refinement process, the evolving mesh of the offcenter refinement looks worse than that of the circumcenter refinement. This can be fixed by changing the bad triangle handling order.

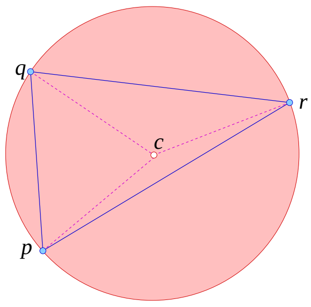

Figure 2: *When $\alpha > 30^\circ$, circumcenter insertion introduces shorter pairwise distances than existing ones, i.e. $|pc| = |qc| = |rc| < |pq|$*

2.1.2. With or without (small input angle) constraints.

Prior to our work termination problem was addressed in the context of small input angles [MPW05, PW05, She00]. The reader is referred to [MPW05] which emphasizes the challenge in addressing the termination problem. It is important to note that the the termination problem exists regardless of the simplicity of the input domain. See Figure 4, where the input consists of a pair of points enclosed by a large box. In general, for any input domain, it is common to observe overrefinement occurring in regions far away from the input features whenever α is large. In this paper, we do not limit ourselves to certain type of constraints and rather address the termination problem in general. Our analysis in Section 4 focuses on point sets. However, this work can be easily

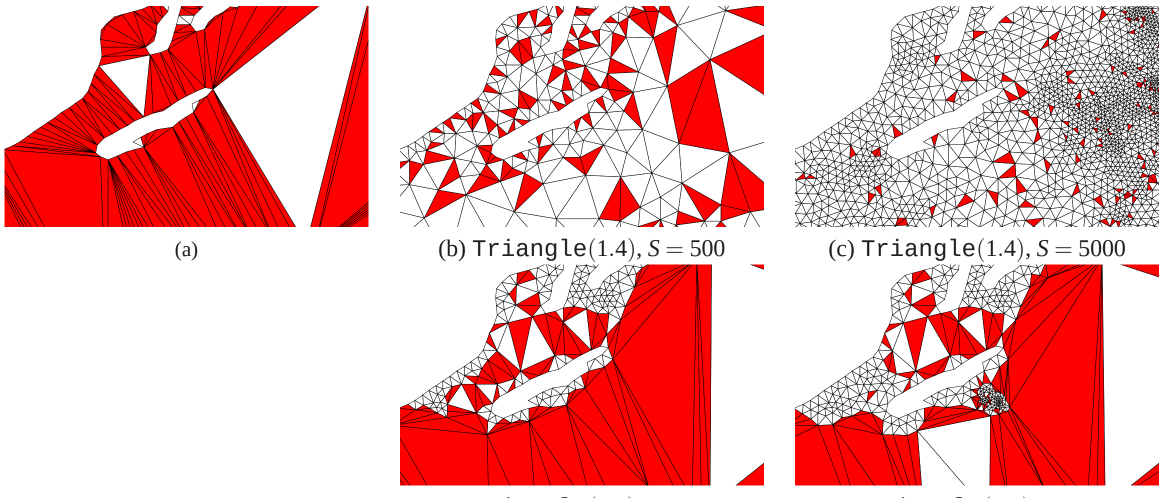
(a)
(b) Triangle(1.4), $S = 500$
(c) Triangle(1.4), $S = 5000$
(d) Triangle(1.6), $S = 500$
(e) Triangle(1.6), $S = 750$

Figure 3: *Termination problem shown for the constraint angle $\alpha = 35.5^\circ$ on the Lake Superior data set (zoomed in). Bad triangles are shaded (red). (a) Delaunay triangulation of the input. (b) and (c) Evolving triangulation using the circumcenter insertion. (d) and (e) Evolving triangulation using the offcenter insertion. Neither of the two algorithms terminates hence interrupted after inserting S Steiner points.*

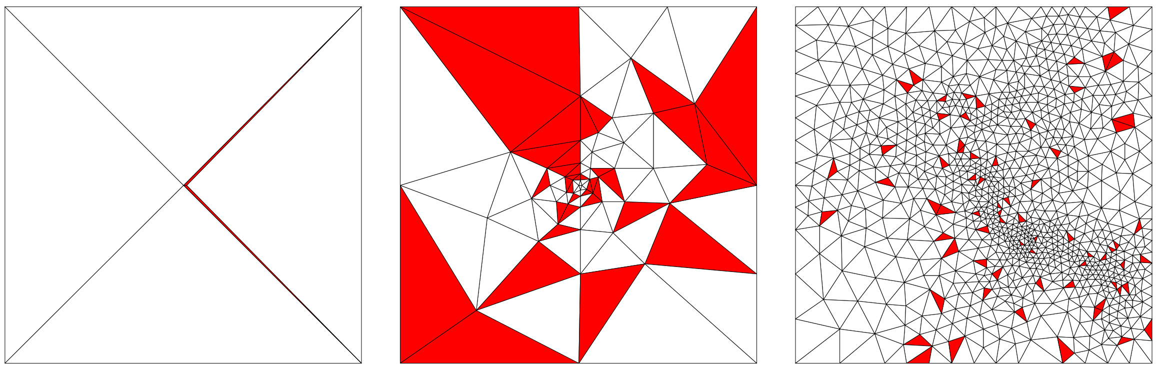

Figure 4: *Termination problem is shown for $\alpha = 35^\circ$ on a simple data set: a pair of points at three unit distance from each other inside a box of side length 100 units. Bad triangles are shaded (red). (a) Delaunay triangulation of the input. (b) and (c) Evolving triangulation using the circumcenter insertion. The refinement process does not terminate hence interrupted after inserting S Steiner points.*

coupled with others considering general planar straight line graphs. Our experiments, on data sets that are planar straight line graphs with small angles verify this.

2.2. Mesh Size

The output size is crucial for the efficiency of the methods using these meshes in various applications. Also, the smaller the number of Steiner points the faster the refinement algorithm will be (if the Steiner point computation time is kept to be the same). Simpson [Sim06] recently showed empirical evidence that standard Delaunay refined meshes (through circumcenter insertion) are roughly twice the size of the optimal meshes for an application he calls function approximation. Our results here confirm this study as we get a size improvement of roughly factor two compared to the original method of Ruppert.

3. LOCALLY OPTIMAL STEINER POINTS

Definition 1 Given an ordered pair of points (p,q) in the input domain. Consider the circle that goes through p and q,

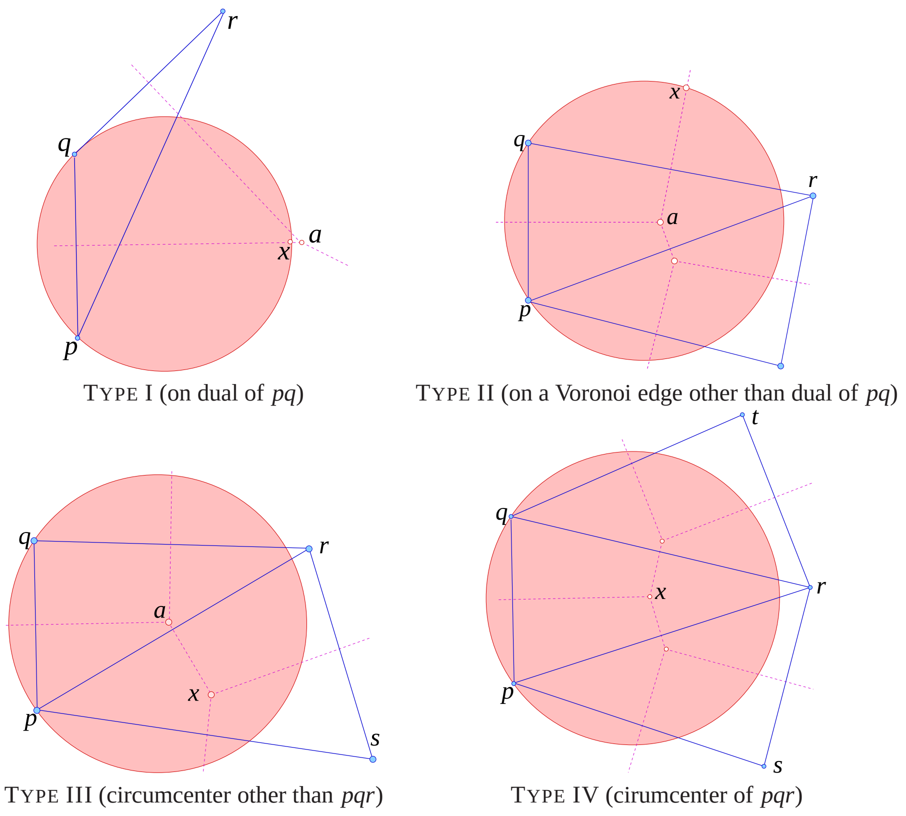
r
q
a
x
p
TYPE I (on dual of *pq*)
x
q
r
a
p
TYPE II (on a Voronoi edge other than dual of *pq*)
q
r
a
x
p
s
TYPE III (circumcenter other than *pqr*)
t
q
r
x
p
s
TYPE IV (cirumcenter of *pqr*)

Figure 5: *Find a point x inside the (shaded disk) petal of pq that is furthest away from all existing vertices. Such a point can be on a Voronoi edge (top) or a Voronoi vertex (bottom).*

of radius $2\sin\alpha|pq|$ whose center is on the right side of the directed edge *pq*. The disk bounded by this circle is called the *petal* of *pq*, denoted by $petal(pq)$.

Note that every edge has two petals (of interest), unless both its endpoints are on the boundary. In a triangulation, petals (of interest) of every edge *pq* must include another vertex. Otherwise, there would be a bad triangle in the triangulation incident to *p* and *q*. HarPeled and Üngör [HPÜ05] proved that the triangulation of a points set is good if and only if petal of every pair of points contains another point. In the following algorithm, we suggest to pick a Steiner point inside empty petals furthest away from all existing vertices.

Algorithm 1

```
Compute the Delaunay triangulation of the input
while $\exists$ a bad triangle $pqr$ with shortest edge $pq$
    insert a point $x \in petal(pq)$ of its shortest edge $pq$
        which is furthest from all existing vertices
```

Note that this point is either a Voronoi vertex (but not necessarily the circumcenter of *pqr*) or on a Voronoi edge. We classify these points into four different types in order to relate them with the existing refinement strategies. This classification also helps us presenting a theoretical and experimental assessment of our method. Let *pqr* be a bad triangle whose shortest edge is *pq*. Then, the point inside the petal of *pq* furthest away from all existing vertices is one of the following types.

- TYPE I. The intersection of the Voronoi dual of *pq* and the boundary of the petal of *pq*. This case happens when the circumcenter of *pqr* is outside the petal of *pq*. This type is nothing but the offcenters described in [Üng04].
- TYPE II. A point on a Voronoi edge other than the dual of the shortest edge *pq*. This is a new type of Steiner point for Delaunay refinement and should be seen as an extension of the offcenter idea to the triangles that are almost good (circumcenter of *pqr* is inside the petal of *pq*) .
- TYPE III. The circumcenter of a nearby triangle to *pqr*. Such a triangle must be acute. So, this type is a sink circumcenter. However, it is not necessarily the sink of the considered bad triangle *pqr* (as is the case in Figure 5). Hence, our use of sinks is somewhat different than the original sink insertion algorithm of Edelsbrunner and Guoy [EG01].

- TYPE IV The circumcenter of the bad triangle pqr.

We compute the location (and also the type) of the Steiner point simply by doing a local search on the Voronoi graph. The angles of the visited triangles provide guidance in this search. For instance, if the smallest angle of the bad triangle is less than $\alpha/2$, then we know for sure that the locally optimal Steiner point is of TYPE I (an offcenter). In general, it is better to start the search visiting the triangle opposite to the largest angle of the bad triangle. We elaborate more on the implementation details of the algorithm in Section 6.

4. ANALYSIS

The analysis of the Delaunay refinement algorithms relies on lower bounds on the local feature size of the Steiner points. As the local feature size of the Steiner points we inserted are not any smaller than that of the circumcenters, it is relatively easy to extend the analysis of the standard refinement techniques here. See for instance [Üng04] for an analysis of the refinement that uses Type I and Type IV Steiner points. One might expect that an improvement on the angle bound. Such an improvement, however, seems difficult to prove and left as an open problem. Here we would like to provide some theoretical evidence on why our strategy works for large values of threshold angle α.

The following lemma is more meaningful in the light of the experimental study as to the percentage use of each type of Steiner points (presented in Section 6).

We further classify the Type II vertices for the sake of analysis. Suppose the Steiner point we used is on the dual edge of qr of a bad triangle with shortest edge pq. A TYPE II Steiner point is called a TYPE II (A) if $\angle qpr \geq \pi/2$, a TYPE II (B) if $\angle pqr \leq \angle qpr < \pi/2$, and a TYPE II (C) otherwise.

Lemma 1 Given a periodic point set Ω and a desired minimum angle α as input, the ALGORITHM 1 does not create any feature shorter than the existing ones while inserting a vertex x of type

(a) TYPE I Steiner points if $\alpha \leq \pi/3 = 60°$.
(b) TYPE II (A) Steiner points if $\alpha \leq \pi/4 = 45°$.
(c) TYPE II (B) Steiner points if $\alpha \leq \pi/5 = 36°$.
(d) TYPE II (C), TYPE III and TYPE IV Steiner points if $\alpha \leq \pi/6 = 30°$.

Proof Let pqr be a bad triangle with shortest edge pq.
(a) Since x is on the Voronoi edge of pq, its nearest neighbors among the existing vertices are p and q. Observing that $|xp| < |pq|$ if and only if $\angle pxq > \pi/3$ completes this part of proof.
(b) Assume, without loss of generality, that x is on the Voronoi dual of qr. So, $\angle qpr$ is a non-acute angle. Let l be the line the line orthogonal to pq and goes through p. (Figure 6 (top).) Let y be the other (than p) intersection of line l and $\partial petal(pq)$. Without loss of generality assume that $petal(pq)$ is unit disk. Then, the length of the arc pq is 2α (in radians). Let x' be the midpoint of the arc yq. This means, the length of the arc $x'q$ is $\pi/2$. Hence, $\alpha \leq \pi/4$ if and only if $pq \leq x'q$. Note that there is no point below the line l and outside the $petal(pq)$ that is closer to x' than q. So, $|x'q| \geq |xq|$. Then, we conclude that $|xq| \geq |pq|$ if $\alpha \leq \pi/4$.
(c) This part of the proof is similar to (b). Assume, without loss of generality, that x is on the Voronoi dual of qr. So, $\angle pqr \leq \angle qpr < \pi/2$. Let l be the line the line orthogonal to pq and goes through the midpoint of pq. (Figure 6 (bottom).) Let y be the intersection point of line l and $\partial petal(pq)$ that is furthest from pq. Without loss of generality assume that $petal(pq)$ is unit disk. Then, the length of the arc pq is 2α (in radians) and the length of the arc yq is $\pi - \alpha$. Let x' be the midpoint of the arc yq. This means, the length of the arc $x'q$ is $\pi/2 - \alpha/2$. Hence, $\alpha \leq \pi/5$ if and only if $pq \leq x'q$. Note that there is no point below the line l and outside the $petal(pq)$ that is closer to x' than q. So, $|x'q| \geq |xq|$. Then, we can conclude that $|xq| \geq |pq|$ if $\alpha \leq \pi/5$.
(d) Straightforward to show. □

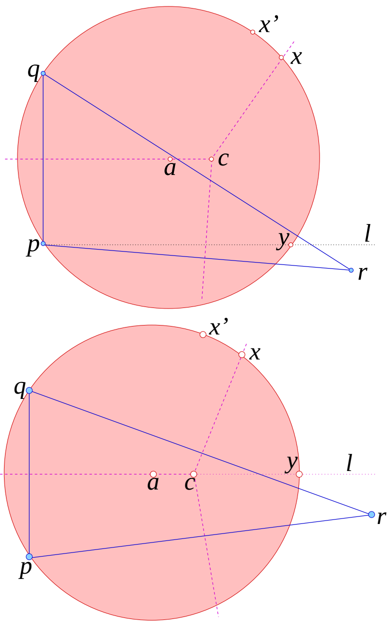

Figure 6: TYPE II (A) AND (B)

Lemma 1 suggests that if we could show that the ALGORITHM 1 only uses, say TYPE I and TYPE II (A) Steiner points, then it would (provably) compute triangulations with minimum angle of 45°. Unfortunately, such a premise does not seem plausable. Our experiments (presented in Section 6) show that all four types are employed by Algorithm 1 in different amounts.

The following theorem follows from the above lemma.

Proof is similar to the results in traditional Delaunay refinement and hence omitted here.

Theorem 1 Given a periodic point set Ω and a desired minimum angle $\alpha \leq 30^\circ$ ALGORITHM 1 terminates with a correct output.

Time Complexity. A straightforward running time analysis leads to a quadratic time complexity bound for ALGORITHM 1 as is the case for most other Delaunay refinement algorithms. However, this bound can be improved to $O(n \log n + m)$, where n is the input size and m is the output size, by using a technique recently introduced by Har-Peled and Üngör [HPÜ05]. This technique employs a balanced quadtree as a data structure and takes advantage of the locality of the Steiner points with respect to the shortest edge of bad triangles. Since the adaptation of the Har-Peled Üngör result is lengthy but somewhat straightforward, we omit it in this short write-up.

5. RELOCATION and REFINEMENT

Based on our analysis (Section 4) and experiments with Algorithm 1, we observed that the refinement process introduces shorter edges than existing ones usually when a bad triangle is almost good and also the neighbor triangles are good or almost good. This suggests that it might be easy to fix such bad triangles by a local smoothing strategy. While one might explore various powerful smoothing strategies [ABE97] in these cases, we opt for simplicity and efficiency. We first recall couple of definitions and then describe a simple adjustment to our algorithm. The *star* of a vertex a consists of all triangles that contain a. The *link* of a, then, consists of all edges of triangles in the star that are disjoint from a. A vertex is said to be *free* if it was inserted by the refinement algorithm as a Steiner point. Input vertices are not free and never relocated. For each bad triangle, we first (one-at-a-time) attempt to relocate its free vertices. If one of its free vertices find a new location so that all the triangles in its (new) star become good, we perform the relocation. Otherwise, we proceed with a new Steiner point insertion as described in ALGORITHM 1.

The standard analysis techniques (for proving termination and output size optimality) that are used for the previous Delaunay refinement algorithms do not apply for ALGORITHM 2 due to the point relocation step. While proving an improved bound for it is left open, we can match the earlier bounds by a simple modification to ALGORITHM 2: apply the relocation only for large values of the constraint angle. This modification keeps ALGORITHM 2 still effective for large constraint angles and provably good for small constraint angles.

6. EXPERIMENTS

We implemented the proposed algorithms and run experiments on various data sets and point distributions. The output triangulations of the new algorithm is shown in Figure 8. Performance plots are similar for various data sets.

Algorithm 2

```
  Compute the Delaunay triangulation ($DelTri$) of the input
  Let $P$ denote the maintained point set
  **while** $\exists$ a bad triangle $pqr$ in $DelTri(P)$
      relocated := FALSE
      **for** each free vertex $a$ of $pqr$
          **if** $\mathcal{K} = \bigcap_{xy \in link(a)} petal(xy) \neq \emptyset$
              **and** $\exists b \in \mathcal{K}$ such that all triangles of $star(b)$
                  in the $DelTri(P \cup \{b\} - \{a\})$ are good
          **then**
              delete $a$; insert $b$; relocated:=TRUE; **break**;
      **endfor**
      **if** relocated == FALSE **then**
          insert a point $x \in petal(pq)$ of its shortest edge $pq$
              which is furthest from all existing vertices
  **endwhile**
```

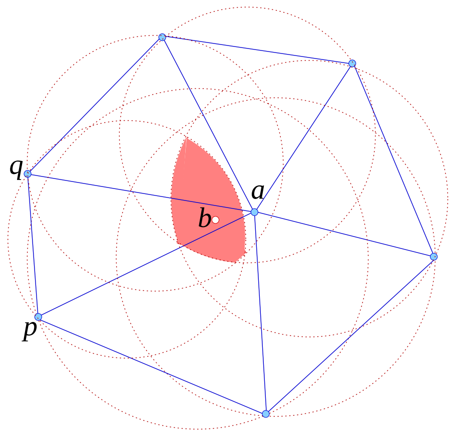

Figure 7: *Relocating a free vertex of a bad triangle pqr to the intersection of the petals of the link of $a = r$.*

6.1. Implementation

Our implementation is a fairly modest modification of the `Triangle` software. Here, we explain the crucial changes. For computing the TYPE II Steiner points, we have implemented a primitive that computes the intersection(s) of a ray and a circle. This computation is slightly more expensive than computing circumcenters and is common to graphics software [MST89]. A similar primitive that computes the intersections of two circles is implemented for the relocation step of ALGORITHM 2. In this step, however we avoid computing the exact intersection of all petals on the link. Instead, we use a simple sampling strategy which proves effective and efficient. For each pair of petals, we pick a sample of points on the line segment connecting intersection points of

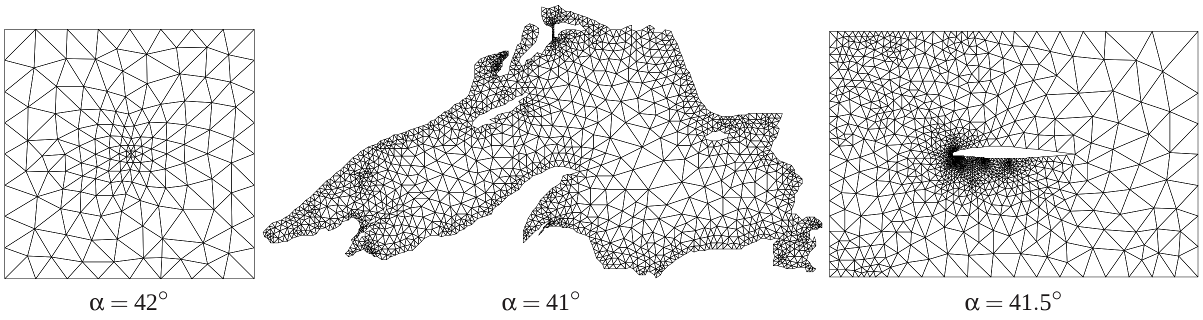

Figure 8: *Output of the* ALGORITHM 2 *for various data sets for various large* α *values.*

the two petals. We enumerate the petal pairs starting from those that are furthest from each other on the link. If there exist any two petal pair with no intersection, then we terminate the process and conclude that there is no relocation point. Otherwise, we test for each sampling point whether it forms a good triangle with each of the edges on the link. Alternative strategies for point relocation can be explored.

6.2. Data Sets.

We ran our experiments on various data sets including the following:

1. *Lake Superior* consists of 522 points, 522 segments some of which meet at small angles, and 7 holes.
2. *Boxed Point Pair* consists of six points two of which are located unit distance from each other and at the center of a square of side length 100 units.
3. *Boeing* consists of 30 points and 30 segments and a hole modeling an airplane wing. The sampling around the tip of the wing is very fine (to allow accurate simulations).
4. *Random Points* consists of randomly spreaded $1,000$ points inside a square box.
5. *Turkey* consists of 216 points and 216 segments and two components.

6.3. Performance Comparison

Here we present a comparative study of our algorithms with the previous Delaunay refinement implementations on three performance measures: angle threshold, output size, and running time.

Angle threshold. Plots in Figure 11 presents a comparison of our approach to the standard Delaunay refinement algorithms. All plots agree that the standard Delaunay refinement algorithm is impotent for angles larger than $\alpha = 35^\circ$, where as the new algorithm terminates with correct output for constraint angles up to $\alpha = 42^\circ$.

Output size. The number of triangles in a triangulation is a simple linear function of the number of points in it. Hence, the plots in Figure 11 reflect on the number of triangles in the output. We observed significant improvement on the output size of the refinement algorithm. This improvement is particularly impressive when the threshold angle is large. (See Figure 9 also.)

Running time. The primitives we use for computing the proposed locally optimal Steiner points is slightly more expensive than those used for computing circumcenters. However, we insert fewer Steiner points. Overall, ALLGORITHM 1 runs slightly faster than the previous algorithms, whereas ALLGORITHM 2 has comparable running time. (See Figure 10.) Once we optimize our code, we expect more significant speed up on both algorithms.

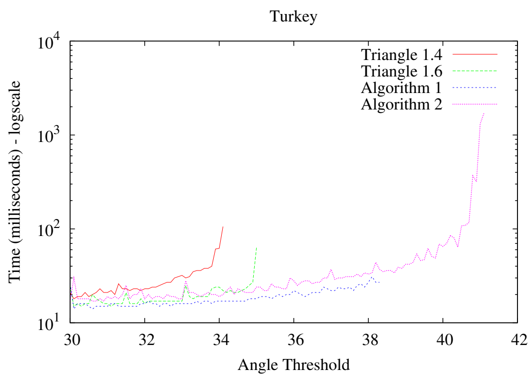

Figure 10: *Plot of running time vs. the constraint angle* α *for the Turkey data set shown in Figure 1.*

7. DISCUSSIONS

Note that the termination and size complexity bounds given for the previous Delaunay refinement apply here for small values of α, as we are more cautious in introducing short features. It would be interesting to prove the same theoretical (termination and size-optimality) bounds for $\alpha > 30^\circ$. It would be also interesting to further improve the practical performance angle bounds say for $\alpha \geq 45^\circ$. This would imply non-obtuse or acute angle triangulations. A natural idea

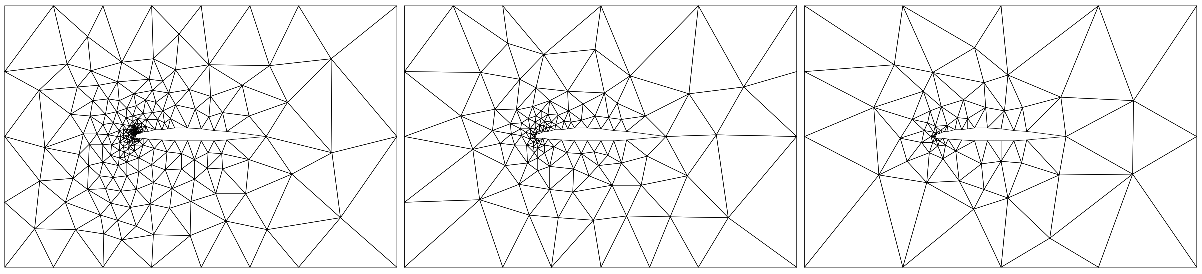

Figure 9: *Output size comparison on the Boeing data set. For* $\alpha = 30^\circ$*, the new algorithm inserts* 62 *Steiner points, almost half as many as the* 118 *Steiner points inserted by the offcenter algorithm which is in turn half as many as the* 236 *Steiner points used by the circumcenter insertion algorithm.*

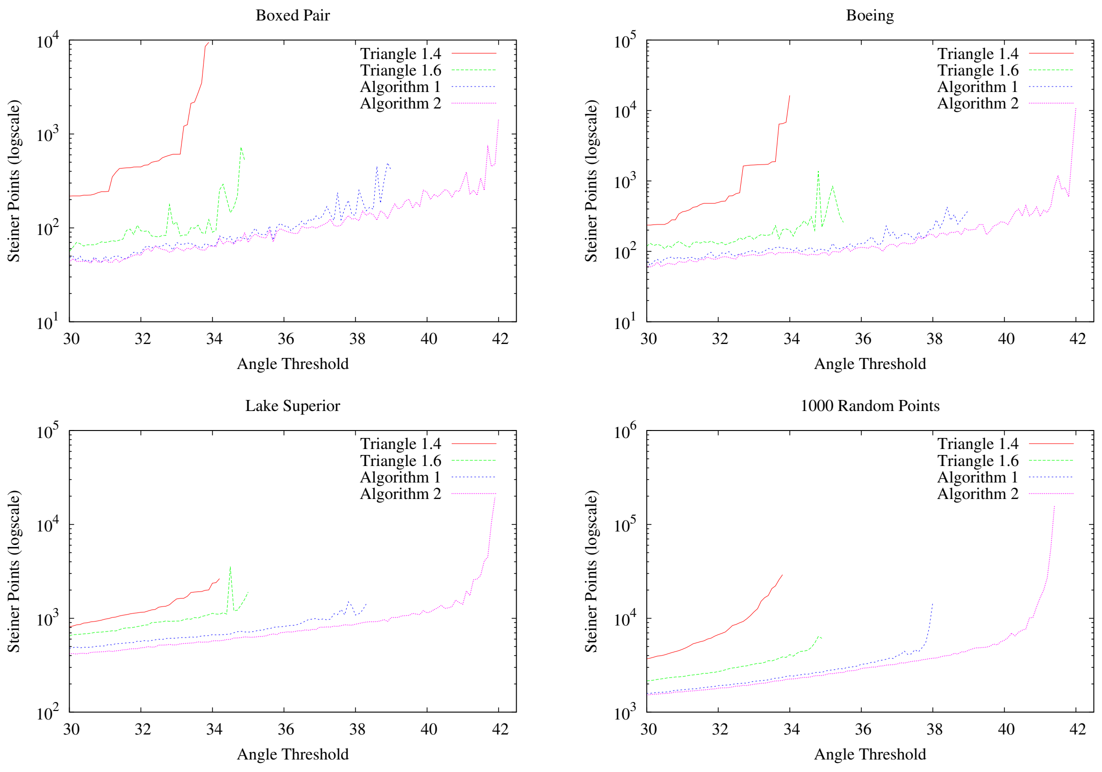

Figure 11: *Plot of the number of Steiner points vs. the constraint angle* α*.*

would be using a lattice guided Delaunay refinement, much like what Labelle suggested in three dimensions [Lab06]. This algorithm essentially is a quadtree/octree type refinement integrated with Delaunay based methods. Our experiments show that while for certain input data we can compute meshes with angles as large as 45°, this idea leads to meshes that are too large to use in practice. For instance, for the range of $30 \leq \alpha \leq 42^\circ$, lattice based refinement gives meshes that are several orders of magnitude larger than the output of our algorithm. Delaunay refinement is a popular technique for computing surface triangulations [Dey06]. We foresee that our algorithms can be easily extended to compute surface meshes. Extension of method to three dimensions is also a natural research direction and is currently under study [JÜ].

	Circumcenter	Offcenter		Algorithm 1				Algorithm 2				
Data Set α	TypeIV	TypeI	TypeIV	TypeI	TypeII	TypeIII	TypeIV	TypeI	TypeII	TypeIII	TypeIV	Relocation
Superior 30	803	182	455	136	152	176	7	145	133	127	6	76
Superior 34	2350	323	787	185	228	237	10	193	203	164	5	79
Superior 38	∞	∞	∞	269	379	381	24	254	316	253	14	101
Superior 41	∞	∞	∞	∞	∞	∞	∞	384	567	331	68	206
Boeing 30	233	59	103	34	42	12	0	39	29	8	0	7
Boeing 34	16309	88	154	50	57	20	3	48	55	12	1	5
Boeing 41	∞	∞	∞	∞	∞	∞	∞	95	152	47	8	47
Random 30	3519	593	1380	445	426	508	30	446	421	458	24	58
Random 34	∞	1076	2770	677	666	802	70	675	644	672	43	141
Random 41	∞	∞	∞	∞	∞	∞	∞	3853	5857	5038	1344	2744

Table 1: *Number of different type of Steiner points used by four different Delaunay refinement algorithms.*

Acknowledgements

The second author thanks to Sariel Har-Peled for the encouragement and technical discussions that lead to this research. This research is conducted with no external funding. More importantly no animals were harmed in the process.

References

[ABE97] AMENTA N., BERN M. W., EPPSTEIN D.: Optimal point placement for mesh smoothing. In *Proc. ACM-SIAM Symp. Discrete Algorithms* (1997), pp. 528–537.

[BE92] BERN M., EPPSTEIN D.: Mesh generation and optimal triangulation. *Computing in Euclidean Geometry* (1992), 23–90.

[BEG94] BERN M., EPPSTEIN D., GILBERT J.: Provably good mesh generation. *J. Comp. System Sciences 48* (1994), 384–409.

[Che89] CHEW L.: *Guaranteed-quality triangular meshes*. Tech. Rep. TR-89-983, Cornell Univ., 1989.

[Dey06] DEY T.: *Curve and Surface Reconstruction: Algorithms with Mathematical Analysis*. Cambridge Univ. Press, 2006.

[Ede01] EDELSBRUNNER H.: *Geometry and topology for mesh generation*. Cambridge Univ. Press, 2001.

[EG01] EDELSBRUNNER H., GUOY D.: Sink insertion for mesh improvement. In *Proc. 17th ACM Symp. Comp. Geom.* (2001), pp. 115–123.

[HPÜ05] HAR-PELED S., ÜNGÖR A.: A time-optimal Delaunay refinement algorithm in two dimensions. In *ACM Symp. on Comp. Geometry* (2005), pp. 228–236.

[JÜ] JAMPANI R., ÜNGÖR A.: A greedy tetrahedal mesh refinement algorithm. (in preparation).

[Lab06] LABELLE F.: Sliver removal by lattice refinement. In *Proc. ACM Symp. on Comp. Geometry* (2006), pp. 347–356.

[MPW05] MILLER G. L., PAV S. E., WALKINGTON N.: When and why delaunay refinement algorithms work. *Int. J. Comput. Geometry Appl. 15*, 1 (2005), 25–54.

[MST89] MIDDLEDITCH A., STACEY T., TOR S.: Intersection algorithms for lines and circles. *ACM Trans. Graph. 8* (1989), 25–40.

[PW05] PAV S., WALKINGTON N.: Delaunay refinement with corner lopping. In *Proc. Int. Meshing Roundtable* (2005).

[RH99] RIVARA M., HITSCHFELD N.: LEPP-Delaunay algorithm: a robust tool for producing size-optimal quality triangulations. In *Proc. 8th Int. Meshing Roundtable* (1999), pp. 205–220.

[Rup93] RUPPERT J.: A new and simple algorithm for quality 2-dimensional mesh generation. In *Proc. ACM-SIAM Symp. on Disc. Algorithms* (1993), pp. 83–92.

[She96] SHEWCHUK J. R.: Triangle: Engineering a 2d quality mesh generator and Delaunay triangulator. In *Proc. First Workshop on Applied Computational Geometry* (1996), pp. 124–133.

[She00] SHEWCHUK J. R.: Mesh generation for domains with small angles. In *Proc. 16th Annual ACM Symposium on Computational Geometry* (2000), pp. 1–10.

[She02] SHEWCHUK J. R.: Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications 22*, 1–3 (2002), 21–74.

[Sim06] SIMPSON B.: How efficient are Delaunay refined meshes? an empirical study. In *Proc.* 15*th Int. Meshing Roundtable* (2006), pp. 215–237.

[TW00] TENG S.-H., WONG C. W.: Unstructured mesh generation: Theory, practice, and perspectives. *Int. J. Computational Geometry & Applications 10*, 3 (Jun 2000), 227–266.

[Üng04] ÜNGÖR A.: Off-centers: A new type of Steiner points for computing size-optimal quality-guaranteed Delaunay triangulations. In *Proc. LATIN* (2004), pp. 152–161.