A Template for Developing Next Generation Parallel Delaunay Refinement Methods

Andrey N. Chernikov and Nikos P. Chrisochoides

Department of Computer Science

College of William and Mary

Williamsburg, Virginia 23185

{ancher,nikos}@cs.wm.edu.

June 17, 2010

Abstract

We describe a complete solution for both sequential and parallel construction of guaranteed quality Delaunay meshes for general two-dimensional geometries. We generalize the existing sequential point placement strategies for guaranteed quality mesh refinement: instead of a specific position for a new point, we derive two types of two-dimensional regions which we call selection disks. Both types of selection disks are inside the circumdisk of a poor quality triangle, with the Type I disk always inside the Type II disk. We prove that any point placement algorithm which inserts Steiner points inside selection disks of Type I terminates, and any algorithm which inserts Steiner points inside selection disks of Type II produces an asymptotically size-optimal mesh. In the area of parallel Delaunay mesh refinement, we develop a new theoretical framework for the construction of graded meshes on parallel architectures, i.e., for parallel mesh generation with element size controlled by a user-defined criterion. Our sufficient conditions of point Delaunay-independence allow to select points for concurrent insertion in such a way that the standard sequential guaranteed quality Delaunay refinement procedures can be applied in parallel to attain the required element quality constraints. Finally, we present a novel parallel algorithm which can be used in conjunction with any sequential point

placement strategy that chooses points within the selection disks. We implemented our algorithm for shared memory multi-core architectures and present the experimental results. Our data show that even on workstations with a few cores, which are now in common use, our implementation is significantly faster than the best sequential counterpart.

Keywords: Delaunay triangulation, mesh generation, parallel refinement

1 Introduction

Next generation guaranteed quality mesh refinement methods will have to be flexible in order to accommodate new challenging applications like medical image analysis and take advantage of current and emerging hardware architectures like multi-core processors. In this paper we lay out an approach for the implementation of next generation parallel guaranteed quality Delaunay mesh generation and refinement. We present our theory and experimental evaluation in two dimensions; three-dimensional results for sequential Generalized Delaunay Refinement and Parallel Delaunay Refinement published in [14] and [15], respectively. The contribution of this paper is the creation of a unified (Parallel and Generalized) Delaunay Refinement method. The unique characteristic of the new framework is its ability to incorporate many different point insertion strategies without any modification both in theory and software, for their parallelization in two and three-dimensions.

Delaunay refinement is a popular technique for generating triangular and tetrahedral meshes for use in the finite element method, the finite volume method, and interpolation in various numeric computing areas. Among the reasons of its popularity is the amenability of the method to rigorous mathematical analysis, which allows to derive guarantees on the quality of the elements in terms of circumradius-to-shortest edge ratio, the gradation of the mesh, and the termination of the algorithm. The parallelization of two-dimensional mesh generation algorithms is particularly important for some three-dimensional simulations which use multiple two-dimensional meshes in different coordinate systems. Some examples include direct numerical simulations of turbulence in cylinder flows with very high Reynolds numbers, see [23], and coastal ocean modeling for predicting storm surge and beach erosion in real-time, see [51]. For the modeling of turbulence, as shown by Karniadakis and Orszag [35], with the increase of the Reynolds number Re, the size of the mesh grows in the order of

$Re^{9/4}$, which motivates the use of parallel mesh generation algorithms. At the same time, the size of the mesh should be as small as possible given the required element quality constraints, which can be attained by using a nonuniform (graded) mesh.

In this paper we address theoretical and practical aspects for the development of both sequential and parallel Delaunay mesh generation algorithms and software that satisfy the following requirements:

1. guarantee well-shaped elements with bounded minimal angle;
2. produce graded meshes, i.e., meshes with element size specified by a user-defined function;
3. offer proofs of termination and size optimality;
4. allow to use custom point placement strategies (e.g., circumcenter, off-center, strategies that satisfy various application-specific criteria, etc.);
5. replace the solution of a difficult domain decomposition problem with an easier data distribution approach without relying on the speculative execution model of an earlier implementation [19, 44];
6. offer performance improvement over the best available sequential software, even on workstations with just a few hardware cores.

The field of sequential guaranteed quality Delaunay refinement has been extensively studied, see [16, 17, 27, 41, 45, 47] and the references therein. However, new ideas and improvements keep being introduced. One of the basic questions is where to insert additional (so-called Steiner) points into an existing mesh in order to improve the quality of the elements. Frey's [26], Ruppert's [45], and early Chew's [17] algorithms use circumcenters of poor quality triangles. Later, Chew [18] suggested to use randomized insertion of *near-circumcenter* points for three-dimensional Delaunay refinement, with the goal of avoiding slivers.

Li and Teng [37, 38] extended the work in [18] by defining a picking sphere with a radius which is a constant multiple of the circumradius of the element. They use two different rules for eliminating the elements with large radius-edge ratio and for eliminating the slivers. In particular, in [37] the

rules are defined as follows: "Add the circumcenter c_τ of any d-simplex with a large $\rho(\tau)$" and "For a sliver-simplex τ, add a good point $p \in \mathcal{P}(\tau)$", where $\rho(\tau)$ is the radius-edge ratio, $\mathcal{P}(\tau)$ is the picking region of simplex τ, and the good point is found by a constant number of random probes. The authors in [37] prove that their algorithm terminates and produces a well graded mesh with good radius-edge ratio and without slivers.

In [14] we introduced Type II selection disks which are defined similarly to the picking region in [37]. We also extended the proofs in [37] to show that any point (not only the circumcenter) from the selection disk (picking region) can be used to eliminate the elements with large radius-edge ratios. We did not address the problem of sliver elimination, however, our work can be used in conjunction with the sliver removal procedure from [37] such that the Delaunay refinement algorithm can choose any points from the selection disks (picking regions) throughout both the stage of the construction of a good radius-edge ratio mesh ("almost good mesh" [37]) and the stage of sliver removal. Intuitively, the requirement of good grading has to be more restrictive than the requirement of termination only, and, therefore, the definitions of the selection disk has to be different to satisfy each of these goals. In [14] we improved upon our previous result [12] by decoupling the definitions of the selection disk used for the proof of termination (Type I) and the selection disk used for the proof of good grading (Type II). As can be seen further in the paper, the selection disk of Type II is always inside the selection disk of Type I of the same element, and as the radius-edge ratio ρ of an element approaches the upper bound $\bar{\rho}$, the Type II disk approaches the Type I disk.

Recently, Üngör [30, 50] proposed to insert specially chosen points which he calls *off-centers*; this method allows to produce smaller meshes in practice and it was implemented in the popular sequential mesh generation software `Triangle` [46]. We expect that other optimization techniques can be used to select positions for new points. Indeed, in [12] we gave an example of a point placement strategy which in some cases allows to achieve even smaller meshes than the off-center method, albeit at significant computation cost. Since one would not like to redesign the parallel algorithm and software to accommodate each of the point placement techniques, in this paper we generalize the sequential Delaunay refinement approaches and develop a framework which allows to use custom point selection strategies.

One of the important applications of the flexibility offered by the use of selection disks is in

conforming the mesh to the boundary between different materials. The advantages are especially pronounced in medical imaging, when the boundaries between different tissues are blurred, see Fig. 1(left). In this case, after the image is segmented, instead of a clear separation, we have a boundary zone of some none-negligible width, see Fig. 1(right). Then the goal of the mesh generation step is to avoid creating edges that would intersect the boundary, which can be achieved by inserting Steiner points inside the boundary zone.

The domain decomposition problem for parallel mesh generation is formulated as follows [20, 39, 40]. Given a domain $\Omega \subset \mathbb{R}^n$, construct the separators $S_{ij} \subset \mathbb{R}^{n-1}$, such that the domain is decomposed into *subdomains* Ω_i:

$$\Omega = \bigcup_{i=1}^{N} \Omega_i, \quad \partial\Omega_i \cap \partial\Omega_j = S_{ij}, \quad i, j = 1, \ldots, N, \quad i \neq j,$$

where $\partial\Omega_i$ is the boundary of subdomain i, while the separators do not create very small angles and other features that will force the degradation of the mesh quality. Linardakis and Chrisochoides [39, 40] described a Medial Axis Domain Decomposition Method for two-dimensional geometries. However, the solution is based on the Medial Axis Transform [2, 21, 29] which is very difficult and expensive to construct for three-dimensional geometries. The approach developed in this paper is domain decomposition independent, i.e., it does not require an explicit construction of internal boundaries between the subdomains which will be forced into the final mesh.

Another approach is to partition and refine the mesh simultaneously, such that when a conflict is detected between concurrently inserted points, some of the point insertions are canceled, which leads to high computation and communication costs. Nave, Chrisochoides, and Chew [19, 44] presented a practical provably-good parallel mesh refinement algorithm for polyhedral domains. This approach allows rollbacks to occur whenever the simultaneously inserted points can potentially lead to an invalid mesh. In the present paper, we develop a theoretical framework which allows us to guarantee *a priori* that concurrently inserted points are Delaunay-independent, and, thus, to avoid computation- and communication-wise expensive rollbacks. By using an auxiliary quadtree data structure, we prove that the points introduced in certain regions of Ω, that correspond to separated leaves of the quadtree, do not have any dependences and can be inserted concurrently.

In addition to parallel mesh generation methods, there is a class of parallel triangulation methods. While the mesh generation problem deals with the selection of points which are not given in the input to achieve required mesh quality, the problem of triangulation is to construct a mesh for a pre-defined point set. A streaming approach to the triangulation problem was implemented by Isenburg, Liu, Shewchuk, and Snoeyink [31]. They achieve large performance gains by using a spatial finalization technique and manage to compute a billion triangle mesh from 500 million points of LIDAR data on a laptop in 48 minutes. A divide-and-conquer projection-based parallel Delaunay triangulation algorithm was developed by Blelloch, Hardwick, Miller, and Talmor [4, 5]. The work by Kadow and Walkington [32–34] extended the work of Blelloch *et al.* for parallel mesh generation and further eliminated the sequential step for constructing an initial mesh, however, all potential conflicts among concurrently inserted points are resolved sequentially through global synchronization which in [33] is implemented by running a dedicated processor. A more extensive review of parallel mesh generation methods can be found in the survey [20].

In our previous work [10, 13] we presented a theoretical framework and the experimental evaluation of a parallel algorithm for constructing guaranteed quality Delaunay meshes which have uniform element size. We proved a sufficient condition of Delaunay-independence, which is based on a relation of the distance between points and the global circumradius upper bound, and which can be verified very efficiently. We also showed that a coarse-grained mesh decomposition can be used in order to guarantee *a priori* that the points in separated regions will be Delaunay-independent. In [11] we developed a theoretical foundation for the parallel construction of non-uniform (graded) meshes. We introduced new, local point independence conditions, and proved that a dynamically constructed quadtree with leaf size reflecting the local mesh density can be used to select Delaunay-independent points. In [12, 14] we generalized the existing point placement techniques and presented the experimental evaluation of our parallel algorithm. In this paper, we unify our previous theoretical and practical results for the parallel construction of graded meshes into a complete solution and study the flexibility–gradation tradeoff of the generalized Delaunay refinement algorithm.

In Section 2 we describe the sequential Generalized Delaunay Refinement (GDR) algorithm. We define the selection disks for the insertion of Steiner points, and present the proofs of termination and size optimality. We introduce the concept of a δ-graded Delaunay refinement algorithm and show

how the constants involved in the proof of size optimality depend on δ. We also give an example of a point selection strategy which allows to reduce the number of inserted Steiner points. Then, in Section 3 we develop local Delaunay-independence conditions and show how quadtree leaves can be used to select subsets of candidate Steiner points for concurrent insertion; we describe our Parallel Generalized Delaunay Refinement (PGDR) algorithm, the implementation details, and the experimental results. Section 4 concludes the paper.

2 Sequential Generalized Delaunay Refinement

2.1 Delaunay refinement background

Let the mesh $\mathcal{M} = (V, T, S)$ consist of a set $V = \{p_i\}$ of vertices, a set $T = \{t_i = \triangle p_u p_v p_w \mid p_u, p_v, p_w \in V\}$ of triangles, and a set $S = \{s_i = p_u p_v \mid p_u, p_v \in V\}$ of constrained segments. We will denote an edge of a triangle as $e(p_i p_j)$, and a straight line segment connecting free points p_i and p_j as $\mathcal{L}(p_i p_j)$. The input to a planar triangular mesh generation algorithm includes a description of *domain* $\Omega \subset \mathbb{R}^2$, which is permitted to contain holes or have more than one connected component. We use a *Planar Straight Line Graph* (PSLG) [46] to delimit Ω from the rest of the plane. Each segment in the PSLG is considered *constrained* and must appear (possibly as a union of smaller subsegments) in the final mesh. The vertices of the PSLG are a subset of the final set of vertices in the mesh.

There are two commonly used parameters that control the quality of mesh elements: an upper bound on the circumradius-to-shortest edge ratio (which is equivalent to a lower bound on a minimal angle [42]) and an upper bound on the element area. We will denote the circumradius-to-shortest edge ratio of triangle t as $\rho(t)$ and the area of triangle t as $\mathcal{A}(t)$. The former upper bound is usually fixed and given by a constant value $\bar{\rho}$, while the latter can vary and be controlled by some user-defined grading function $\bar{\mathcal{A}}(\cdot)$, which can be defined either over the set of triangles or over Ω, depending on the implementation.

Let us call the open disk corresponding to a triangle's circumscribed circle its *circumdisk*. We will use symbols $\bigcirc(t)$ and $r(t)$ to represent the circumdisk and the circumradius of triangle t, respectively. A mesh is said to satisfy the *Delaunay property* if the circumdisk of every triangle does

not contain any of the mesh vertices [22, 24, 27, 47].

Delaunay mesh generation algorithms start with the construction of the initial mesh, which conforms to the input PSLG, and then refine this mesh until the element quality constraints are met. In this paper, we focus on parallelizing the Delaunay refinement stage, which is usually the most memory- and computation-expensive [10]. The general idea of Delaunay refinement is to insert additional (Steiner) points inside the circumdisks of poor quality triangles, which causes these triangles to be destroyed, until they are gradually eliminated and replaced by better quality triangles.

We will extensively use the notion of *cavity* [27] which is the set of triangles in the mesh whose circumdisks include a given point p_i. We will denote $\mathcal{C}_{\mathcal{M}}(p_i)$ to be the cavity of p_i with respect to mesh $\mathcal{M}$ and $\partial\mathcal{C}_{\mathcal{M}}(p_i)$ to be the set of edges which belong to only one triangle in $\mathcal{C}_{\mathcal{M}}(p_i)$, i.e., external edges. When $\mathcal{M}$ is clear from the context, we will omit the subscript. For our analysis, we will use the Bowyer-Watson (B-W) point insertion algorithm [6, 52], which can be written as

$$\begin{aligned} V' &\leftarrow V \cup \{p_i\}, \\ T' &\leftarrow T \setminus \mathcal{C}(p_i) \cup \{\triangle p_i p_j p_k \mid e(p_j p_k) \in \partial\mathcal{C}(p_i)\}, \end{aligned} \tag{1}$$

where $\mathcal{M} = (V, T, S)$ and $\mathcal{M}' = (V', T', S')$ represent the mesh before and after the insertion of p_i, respectively. The set of newly created triangles forms a *ball* [27] of point p_i (denoted $\mathcal{B}(p_i)$), which is the set of triangles in the mesh that have p_i as a vertex.

Sequential Delaunay algorithms treat *constrained* segments differently from triangle edges [45, 47]. A vertex p is said to *encroach upon* a segment s, if it lies within the open diametral disk of s [45]. When a new point is about to be inserted and it happens to encroach upon a constrained segment s, another point is inserted in the middle of s instead [45], and a cavity of the segment's midpoint is constructed and triangulated according to (1).

The proofs of termination and size optimality of Delaunay refinement algorithms [45, 47] explore the relationships between the insertion radius of a point and that of its parent. The *insertion radius* $R(p)$ of point p is defined as the length of the shortest edge connected to p immediately after p is inserted into the mesh [47]. The *parent* $\hat{p}$ of point p is the vertex which is "responsible" for the insertion of p [47]. In particular, if p is inserted inside the circumdisk of a poor quality triangle, $\hat{p}$ is the most recently inserted vertex of the shortest edge of that triangle. If p is a midpoint of an

encroached segment, $\hat{p}$ is the point (possibly rejected for insertion) that encroaches upon that segment. If p is an input vertex, it has no parent. In addition, the proofs require that $\bar{\rho} \geq \sqrt{2}$.

The local feature size function $\mathrm{lfs} : \mathbb{R}^2 \to \mathbb{R}$ for a given point p is equal to the radius of the smallest disk centered at p that intersects two non-incident vertices or segments of PSLG $\mathcal{X}$ [45]. $\mathrm{lfs}\,(p)$ satisfies the Lipschitz condition:

Lemma 1 (Lemma 1 in Ruppert [45], Lemma 2 in Shewchuk [47]) *Given any PSLG $\mathcal{X}$ and any two points p_i and p_j in the plane, the following inequality holds:*

$$\mathrm{lfs}\,(p_i) \leq \mathrm{lfs}\,(p_j) + \|p_i - p_j\| \tag{2}$$

Remark 1 *As shown in [47], if p is an input vertex, then $R\,(p) \geq \mathrm{lfs}\,(p)$. Indeed, from the definition of* $\mathrm{lfs}\,(p)$*, the second feature (in addition to p) which intersects the disk centered at p is either a constrained segment, a constrained facet, or the nearest input vertex visible from p.*

2.2 Generalized Delaunay refinement using selection disks

In this section we introduce two types of selection disks which can be used for the insertion of Steiner points in two dimensions. First, we prove the termination of a Delaunay refinement algorithm with the Type I selection disks. Then we give an example of an optimization based strategy for the insertion of Steiner points from the Type I selection disks which, for small angle bounds, allows to decrease the size of the final mesh in practice. Finally, we introduce the Type II selection disk (which is always inside the Type I selection disk of the same skinny triangle) and prove the good grading and the size optimality.

2.2.1 Proof of Termination with Selection Disks of Type I

Definition 1 (Selection disk of Type I) *If t is a poor quality triangle with circumcenter c, shortest edge length l, circumradius r, and circumradius-to-shortest edge ratio $\rho = r/l > \bar{\rho} \geq \sqrt{2}$, then the* Type I selection disk *for the insertion of a Steiner point that would eliminate t is the open disk with center c and radius $r - \sqrt{2}l$.*

For example, in Figure 2(right), $e\,(p_l p_m)$ is the shortest edge of a skinny triangle $\triangle p_k p_l p_m$ and c is its circumcenter. The selection disk of Type I is the shaded disk with center c and radius $r\,(\triangle p_k p_l p_m) - \sqrt{2}\|p_l - p_m\|$.

Further we will prove that any point inside the Type I selection disk of a triangle can be chosen for the elimination of the triangle, and that the generalized Delaunay refinement algorithm which chooses Steiner points inside Type I selection disks terminates.

Remark 2 *Üngör's off-center always lies inside the selection disk of Type I. Consider Figure 2(left). Suppose $\triangle p_k p_l p_m$ is skinny: $\rho\,(\triangle p_k p_l p_m) > \bar{\rho}$. If we insert its circumcenter c, the new triangle $\triangle c p_l p_m$ may also be skinny. In this case, instead of inserting c, Üngör suggests to insert the* off-center *o chosen on the perpendicular bisector of the shortest edge $e\,(p_l p_m)$ in such a way that the new triangle $\triangle o p_l p_m$ will have circumradius-to-shortest edge ratio equal to exactly $\bar{\rho}$, i.e.,*

$$\rho\,(\triangle o p_l p_m) = \bar{\rho}. \tag{3}$$

If a is the circumcenter of $\triangle o p_l p_m$, and b is the midpoint of edge $e\,(p_l p_m)$, then from (3) *and the Pythagorean theorem for $\triangle a b p_l$ we have:*

$$\|a-b\|^2 = \bar{\rho}^2\|p_l - p_m\|^2 - \frac{1}{4}\|p_l - p_m\|^2 = (\bar{\rho}^2 - \frac{1}{4})\|p_l - p_m\|^2,$$

or

$$\|a-b\| = \frac{\sqrt{4\bar{\rho}^2 - 1}}{2}\|p_l - p_m\|. \tag{4}$$

Noting that

$$\|a-o\| = r\,(\triangle o p_l p_m) = \bar{\rho}\|p_l - p_m\|, \tag{5}$$

we have:

$$\begin{aligned}
\|c-o\| &< r\left(\triangle p_k p_l p_m\right) - \|a-b\| - \|a-o\| \\
&= r\left(\triangle p_k p_l p_m\right) - \frac{\sqrt{4\bar{\rho}^2-1}}{2}\|p_l - p_m\| - \bar{\rho}\|p_l - p_m\| && \text{(from (4) and (5))} \\
&= r\left(\triangle p_k p_l p_m\right) - \left(\frac{\sqrt{4\bar{\rho}^2-1}}{2} + \bar{\rho}\right)\|p_l - p_m\| \\
&\leq r\left(\triangle p_k p_l p_m\right) - \left(\frac{\sqrt{7}}{2} + \sqrt{2}\right)\|p_l - p_m\| && \text{(since } \bar{\rho} \geq \sqrt{2}\text{)} \\
&< r\left(\triangle p_k p_l p_m\right) - \sqrt{2}\|p_l - p_m\|,
\end{aligned}$$

which implies that the off-center o is inside the Type I selection disk of triangle $\triangle p_k p_l p_m$.

Remark 3 *As $\rho\left(\triangle p_k p_l p_m\right)$ approaches $\sqrt{2}$, the Type I selection disk shrinks to the circumcenter c of the triangle. If, furthermore, $\rho\left(\triangle p_k p_l p_m\right) \leq \sqrt{2}$, the selection disk vanishes, which coincides with the fact that the triangle $\triangle p_k p_l p_m$ cannot be considered skinny.*

Lemma 2 *If p_i is a vertex of the mesh produced by a Delaunay refinement algorithm which chooses points within Type I selection disks of triangles with circumradius-to-shortest edge ratios greater than $\bar{\rho} \geq \sqrt{2}$, then the following inequality holds:*

$$R\left(p_i\right) \geq C \cdot R\left(\hat{p}_i\right), \tag{6}$$

where C is defined as follows:

(i) $C = \sqrt{2}$ if p_i is a Steiner point chosen within the Type I selection disk of a skinny triangle;

Otherwise, let p_i be the midpoint of subsegment s. Then

(ii) $C = \frac{1}{\sqrt{2}}$ if $\hat{p}_i$ is a Steiner point which encroaches upon s, chosen within the selection disk of a skinny triangle;

(iii) $C = \frac{1}{2\cos\alpha}$ if p_i and $\hat{p}_i$ lie on incident subsegments separated by an angle of α (with $\hat{p}_i$ encroaching upon s), where $45^\circ \leq \alpha \leq 90^\circ$;

(iv) $C = \sin\alpha$ if p_i and $\hat{p}_i$ lie on incident segments separated by an angle of $\alpha \leq 45^\circ$.

If p_i is an input vertex, then

$$R\left(p_i\right) \geq \mathrm{lfs}\left(p_i\right). \tag{7}$$

Proof We need to present new proofs only for cases (i) and (ii), since the proofs for all other cases are independent of the choice of the point within the selection disk and are given in [47].

Case (i) By the definition of a parent vertex, $\hat{p}_i$ is the most recently inserted endpoint of the shortest edge of the triangle; without loss of generality let $\hat{p}_i = p_l$ and $e(p_l p_m)$ be the shortest edge of the skinny triangle $\triangle p_k p_l p_m$, see Figure 2(right). If $e(p_l p_m)$ was the shortest edge among the edges incident upon p_l at the time p_l was inserted into the mesh, then $\|p_l - p_m\| = R(p_l)$ by the definition of the insertion radius; otherwise, $\|p_l - p_m\| \geq R(p_l)$. In either case,

$$\|p_l - p_m\| \geq R(p_l). \tag{8}$$

Now we can derive the relation between the insertion radius of point p_i and the insertion radius of its parent $\hat{p}_i = p_l$:

$$\begin{aligned} R(p_i) &> \sqrt{2}\|p_l - p_m\| && \text{(from Delaunay property and Definition 1)} \\ &\geq \sqrt{2}R(p_l) && \text{(from (8)).} \end{aligned}$$

Hence, $R(p_i) > \sqrt{2}R(\hat{p}_i)$; choose $C = \sqrt{2}$.

Case (ii) Let $\hat{p}_i$ be inside the Type II selection disk of a skinny triangle $\triangle p_k p_l p_m$, such that $\hat{p}_i$ encroaches upon $e(p_u p_v)$, see Figure 5(left). Since the edge $e(p_u p_v)$ is part of the mesh, there must exist some vertex p_w such that p_u, p_v, and p_w form a triangle. Because p_w is outside of the diametral circle of $e(p_u p_v)$, the circumdisk $\bigcirc(\triangle p_u p_v p_w)$ has to include point $\hat{p}_i$. Therefore, if $\hat{p}_i$ were inserted into the mesh, $\triangle p_u p_v p_w$ would be part of the cavity $\mathcal{C}(\hat{p}_i)$ and the edges connecting $\hat{p}_i$ with p_u and p_v would be created. Therefore,

$$\begin{aligned} R(\hat{p}_i) &\leq \min(\|\hat{p}_i - p_u\|, \|\hat{p}_i - p_v\|) && \text{(from the definition of insertion radius)} \\ &< \sqrt{2}\frac{\|p_u - p_v\|}{2} && \text{(because } \hat{p}_i \text{ encroaches upon } e(p_u p_v)) \\ &= \sqrt{2}R(p_i) && \text{(from the definition of insertion radius);} \end{aligned}$$

choose $C = \frac{1}{\sqrt{2}}$.

Figure 3 shows the relationship between the insertion radii of mesh vertices and the insertion radii of their parents. We can see that if Inequality (6) is satisfied then no new edge will be created whose length is smaller than $\frac{1}{\sqrt{2}}$ times the length of some existing edge and the algorithm will eventually terminate because it will run out of space to insert new vertices.

Theorem 1 (Theorem 4 in [47]) *Let* $\text{lfs}_{\min}$ *be the shortest distance between two non-incident entities (vertices or segments) of the input PSLG. Suppose that any two incident segments are separated by an angle of at least* $60°$*, and a triangle is considered to be skinny if its circumradius-to-shortest edge ratio is larger than* $\bar{\rho}$*, where* $\bar{\rho} \geq \sqrt{2}$*. Ruppert's algorithm will terminate with no triangulation edge shorter than* $\text{lfs}_{\min}$.

The proof of this theorem in [47] is based on the result of Lemma 3 in [47], which establishes Inequality (6) in the context of circumcenter point insertion. Otherwise, the proof is independent of the specific position of inserted points. Since we proved (6) in the context of inserting arbitrary points within Type I selection disks, this theorem also holds in this new context.

The size of the Type I selection disks, and, hence, the flexibility in choosing Steiner points depends on the initial model and can vary significantly. Figure 4 depicts the circumradius-to-shortest edge length ratios of skinny triangles being eliminated by inserting Steiner points in their circumcenters, for the pipe, the cylinder flow, and the Chesapeake bay models, as they were refined to meet the $20°$ minimal angle bound. The horizontal axis show the iteration number, such that each insertion of a Steiner point counts as one iteration. The vertical axis correspond to the circumradius-to-shortest edge ratio of the skinny triangle eliminated by each Steiner point. The larger the circumradius-to-shortest ratio of the triangle, the larger its Type I selection disk: if the circumradius-to-shortest edge length ratio of a triangle is ρ, the radius of the Type I selection disk is $r - \sqrt{2}l = r(1 - \sqrt{2}/\rho)$.

2.2.2 An Example of a Point Selection Strategy

Let us consider Figure 5(right). We can see that the off-center o of the skinny triangle $\triangle p_k p_l p_m$ is not the only location for a Steiner point p_i that will lead to the creation of the new triangle incident to the edge $e\,(p_l p_m)$ with circumradius-to-shortest edge ratio equal to exactly $\bar{\rho}$. The arc shown in

bold in the figure is the intersection of the circle passing through p_l, p_m, and o with the selection disk of $\triangle p_k p_l p_m$. Let us denote this arc as Γ. The thin arcs show parts of the circumcircles of other triangles in the mesh. We can observe that the cavity $\mathcal{C}(p_i)$ will vary depending on the location of p_i, according to the set of triangle circumdisks that include p_i. Let us also represent the penalty for deleting triangle t as $P(t)$:

$$P(t) = \begin{cases} -1, & \text{if } (\rho(t) > \bar{\rho}) \vee (\mathcal{A}(t) > \bar{\mathcal{A}}), \\ 1, & \text{otherwise.} \end{cases}$$

Then our objective is to minimize the profit function associated with the insertion of point p_i as the sum of the penalties for deleting all triangles in the cavity $\mathcal{C}(p_i)$:

$$\min_{p_i \in \Gamma} F(p_i), \quad F(p_i) = \sum_{t \in \mathcal{C}(p_i)} P(t)$$

In other words, we try to minimize the number of deleted good quality triangles and at the same time to maximize the number of deleted poor quality triangles. The results of our experiments with the cylinder flow (Figure 6(left)) and the pipe cross-section (Figure 6(right)) models using `Triangle` version 1.6 [46] are summarized in Tables 1 and 2. We do not list the running times since our experimental implementation is built on top of `Triangle`, but we do not take advantage of its efficient routines for our intermediate calculations as do the circumcenter and the off-center point insertion methods. From Table 1 we can see that our optimization-based method tends to produce up to 20% fewer triangles than the circumcenter method and up to 5% fewer triangles than the off-center method for small values of the minimal angle bound and no area bound, and the improvement diminishes as the angle bound increases. Parts of the pipe mesh for the three point insertion methods are also shown in Figure 7. Table 2 lists the results of the similar experiments, with an additional area bound constraint. We observe that the introduction of an area bound effectively voids the difference among the presented point insertion strategies.

2.2.3 Proof of Good Grading and Size Optimality with Selection Disks of Type II

Definition 2 (Selection disk of Type II) *If t is a poor quality triangle with circumcenter c, shortest edge length l, circumradius r, and circumradius-to-shortest edge ratio $\rho = r/l > \bar{\rho} \geq \sqrt{2}$, then the* Type II selection disk *for the insertion of a Steiner point that would eliminate t is the open disk with center c and radius $r(1 - \frac{\sqrt{2}}{\bar{\rho}})$.*

For example, in Figure 8, $e\,(p_l p_m)$ is the shortest edge of a skinny triangle $\triangle p_k p_l p_m$ and c is its circumcenter. The Type II selection disk for this triangle is the shaded disk with center c and radius $r\,(\triangle p_k p_l p_m)\,(1 - \frac{\sqrt{2}}{\bar{\rho}})$.

Remark 4 *Note from Definitions 1 and 2 that the radius of the Type II selection disk is always smaller than the radius of the Type I selection disk of the same skinny triangle because $r(1 - \frac{\sqrt{2}}{\bar{\rho}}) = r - \frac{\rho}{\bar{\rho}}\sqrt{2}l$ and $\rho > \bar{\rho}$.*

Remark 5 *As $\bar{\rho}$ approaches $\sqrt{2}$ the radius of the Type II selection disk approaches zero, which means that the selection disk shrinks to the circumcenter point.*

As can be seen further below, the price which we pay for the gain in the flexibility in choosing points is the increase of the constants which bound the size of the mesh. To classify the Delaunay refinement algorithms with respect to the theoretical bounds on mesh grading we need the following definition.

Definition 3 (δ-graded Delaunay refinement algorithm) *If for every triangle t with circumcenter c, circumradius r, shortest edge length l, and circumradius-to-shortest edge length ratio $\rho = r/l > \bar{\rho} \geq \sqrt{2}$ a Delaunay refinement algorithm selects a Steiner point p_i within the Type II selection disk such that $\|p_i - c\| < r(1 - \delta)$, where*

$$\frac{\sqrt{2}}{\bar{\rho}} \leq \delta \leq 1,$$

we say that this Delaunay refinement algorithm is δ-graded.

The analysis below assumes that all angles in the input PSLG are greater than 45° or 60°. In practice such geometries are rare; however, a modification of the algorithm with a concentric circular

shell splitting [45, 47] allows to guarantee the termination of the algorithm even though the small angles adjacent to the segments of the input PSLG cannot be improved.

Lemma 3 *If p_i is a vertex of the mesh produced by a δ-graded Delaunay refinement algorithm then the following inequality holds:*

$$R(p_i) \geq C \cdot R(\hat{p}_i), \tag{9}$$

where we distinguish among the following cases:

(i) $C = \delta\bar{\rho}$ *if p_i is a Steiner point chosen within the Type II selection disk of a skinny triangle;*

Otherwise, let p_i be the midpoint of subsegment s. Then

(ii) $C = \frac{1}{\sqrt{2}}$ *if $\hat{p}_i$ is a Steiner point which encroaches upon s, chosen within the Type II selection disk of a skinny triangle;*

(iii) $C = \frac{1}{2\cos\alpha}$ *if p_i and $\hat{p}_i$ lie on incident subsegments separated by an angle of α (with $\hat{p}_i$ encroaching upon s), where $45^\circ \leq \alpha \leq 90^\circ$;*

(iv) $C = \sin\alpha$ *if p_i and $\hat{p}_i$ lie on incident segments separated by an angle of $\alpha \leq 45^\circ$.*

If p_i is an input vertex, then

$$R(p_i) \geq \mathrm{lfs}(p_i).$$

Proof We need to present a new proof only for case (i) since the proof for case (ii) is the same as in Lemma 2, and the proofs for all other cases are independent of the choice of the point within the selection disk and are given in [47].

Case (i) As in the proof of Case (i) of Lemma 2, assuming that $e(p_l p_m)$ is the shortest edge of the skinny triangle $\triangle p_k p_l p_m$ and $\hat{p}_i = p_l$, we derive relation (8). Then

$$\begin{aligned}
R(p_i) &> \delta r && \text{(from Delaunay property and Definition 3)} \\
&= \delta\rho\|p_l - p_m\| && \text{(since } \rho = \tfrac{r}{\|p_l - p_m\|}\text{)} \\
&> \delta\bar{\rho}\|p_l - p_m\| && \text{(since } \rho > \bar{\rho}\text{)} \\
&\geq \delta\bar{\rho}R(p_l) && \text{(from (8)).}
\end{aligned}$$

Hence, $R(p_i) > \delta\bar{\rho}R(\hat{p}_i)$; choose $C = \delta\bar{\rho}$.

The quantity $D(p)$ is defined as the ratio of $\mathrm{lfs}(p)$ over $R(p)$ [47]:

$$D(p) = \frac{\mathrm{lfs}(p)}{R(p)}. \tag{10}$$

It reflects the density of vertices near p at the time p is inserted, weighted by the local feature size.

Lemma 4 *If p is a vertex of the mesh produced by a δ-graded Delaunay refinement algorithm and C is the constant specified by Lemma 3, then the following inequality holds:*

$$D(p) \leq \frac{2-\delta}{\delta} + \frac{D(\hat{p})}{C}. \tag{11}$$

Proof If p is inside a Type II selection disk of a skinny triangle with circumradius r, then

$$\begin{aligned}
\|p-\hat{p}\| &< 2r - \delta r && \text{(from the definition of } \hat{p} \text{ and Def. 3)} \\
&= (2-\delta)r \\
&= \tfrac{2-\delta}{\delta}\delta r \\
&< \tfrac{2-\delta}{\delta}R(p) && \text{(from Delaunay property and Def. 3).}
\end{aligned}$$

If p is an input vertex or lies on an encroached segment, then

$$\begin{aligned}
\|p-\hat{p}\| &\leq R(p) && \text{(by definitions of } \hat{p} \text{ and } R(p)\text{)} \\
&\leq \tfrac{2-\delta}{\delta}R(p) && \text{(since from Def. 3, } \delta \leq 1\text{).}
\end{aligned}$$

In all cases,

$$\|p-\hat{p}\| \leq \frac{2-\delta}{\delta}R(p). \tag{12}$$

Then

$$\begin{aligned}
\mathrm{lfs}(p) &\leq \mathrm{lfs}(\hat{p}) + \|p-\hat{p}\| && \text{(from Lemma 1)} \\
&\leq \mathrm{lfs}(\hat{p}) + \tfrac{2-\delta}{\delta}R(p) && \text{(from (12))} \\
&= D(\hat{p})\,R(\hat{p}) + \tfrac{2-\delta}{\delta}R(p) && \text{(from (10))} \\
&\leq D(\hat{p})\,\tfrac{R(p)}{C} + \tfrac{2-\delta}{\delta}R(p) && \text{(from Lemma 3).}
\end{aligned}$$

The result follows from the division of both sides by $R(p)$.

Lemma 5 (Extension of Lemma 7 in [47] and Lemma 2 in [45]) *Suppose that $\bar{\rho} > \sqrt{2}$ and the smallest angle in the input PSLG is strictly greater than 60°. There exist fixed constants C_T and C_S such that, for any vertex p inserted (or considered for insertion and rejected) by a δ-graded Delaunay refinement algorithm, $D(p) \leq C_T$, and for any vertex p inserted at the midpoint of an encroached subsegment, $D(p) \leq C_S$. Hence, the insertion radius of a vertex has a lower bound proportional to its local feature size.*

Proof The proof is by induction and is similar to the proof of Lemma 7 in [47]. The base case covers the input vertices, and the inductive step covers the other two types of vertices: free vertices and subsegment midpoints.

Base case: The lemma is true if p is an input vertex, because in this case, by Remark 1, $D(p) = \text{lfs}(p)/R(p) \leq 1$.

Inductive hypothesis: Assume that the lemma is true for $\hat{p}$, i.e., $D(\hat{p}) \leq \max\{C_T, C_S\}$.

Inductive step: There are two cases:

(i) If p is in the Type II selection disk of a skinny triangle, then

$$\begin{aligned} D(p) &\leq \tfrac{2-\delta}{\delta} + \tfrac{D(\hat{p})}{C} && \text{(from Lemma 4)} \\ &= \tfrac{2-\delta}{\delta} + \tfrac{D(\hat{p})}{\delta\bar{\rho}} && \text{(from Lemma 3)} \\ &\leq \tfrac{2-\delta}{\delta} + \tfrac{\max\{C_T, C_S\}}{\delta\bar{\rho}} && \text{(by the inductive hypothesis).} \end{aligned}$$

It follows that one can prove that $D(p) \leq C_T$ if C_T is chosen so that

$$\frac{2-\delta}{\delta} + \frac{\max\{C_T, C_S\}}{\delta\bar{\rho}} \leq C_T. \tag{13}$$

(ii) If p is the midpoint of a subsegment s such that $\hat{p}$ encroaches upon s, then we have 3 sub-cases:

(ii-a) If $\hat{p}$ is an input vertex, then the disk centered at p and touching $\hat{p}$ has radius less than the radius of the diametral disk of s and therefore $\text{lfs}(p) < R(p)$. Thus, $D(p) < 1$ and the lemma holds.

(ii-b) If $\hat{p}$ is a rejected point from the Type II selection disk of a skinny triangle or lies on a

segment not incident to s, then

$$\begin{aligned} D(p) &\leq \tfrac{2-\delta}{\delta} + \tfrac{D(\hat{p})}{C} && \text{(from Lemma 4)} \\ &= \tfrac{2-\delta}{\delta} + \sqrt{2}D(\hat{p}) && \text{(from Lemma 3)} \\ &\leq \tfrac{2-\delta}{\delta} + \sqrt{2}C_T && \text{(by the inductive hypothesis).} \end{aligned}$$

(ii-c) If $\hat{p}$ lies on a segment incident to s, then

$$\begin{aligned} D(p) &\leq \tfrac{2-\delta}{\delta} + \tfrac{D(\hat{p})}{C} && \text{(from Lemma 4)} \\ &= \tfrac{2-\delta}{\delta} + 2\cos\alpha D(\hat{p}) && \text{(from Lemma 3)} \\ &\leq \tfrac{2-\delta}{\delta} + 2C_S\cos\alpha && \text{(by the inductive hypothesis).} \end{aligned}$$

It follows that one can prove that $D(p) \leq C_S$ if C_S is chosen so that both of the following relations (14) and (15) are satisfied:

$$\frac{2-\delta}{\delta} + \sqrt{2}C_T \leq C_S, \tag{14}$$

and

$$\frac{2-\delta}{\delta} + 2C_S\cos\alpha \leq C_S. \tag{15}$$

If $\delta\bar{\rho} > \sqrt{2}$, relations (13) and (14) can be simultaneously satisfied by choosing

$$C_T = \frac{(2-\delta)(\bar{\rho}+\delta)}{\delta\bar{\rho}-\sqrt{2}}, \quad \text{and} \quad C_S = \frac{(2-\delta)\bar{\rho}(1+\sqrt{2})}{\delta\bar{\rho}-\sqrt{2}}.$$

If the smallest input angle $\alpha_{\min} > 60°$, relations (13) and (15) can be simultaneously satisfied by choosing

$$C_T = \frac{2-\delta}{\delta} + \frac{C_S}{\delta\bar{\rho}}, \quad \text{and} \quad C_S = \frac{2-\delta}{\delta(1-2\cos\alpha_{\min})}.$$

Theorem 2 (Theorem 8 in [47], Theorem 1 in [45]) *For any vertex p of the output mesh, the distance to its nearest neighbor is at least $\frac{\mathrm{lfs}(p)}{C_S+1}$.*

The proof in [47] relies only on Lemmata 1 and 5 here and, therefore, holds for the arbitrary points chosen within selection disks of skinny triangles.

Theorem 2 is used in the proof of the following theorem.

Theorem 3 (Theorem 10 in [47], Theorem 14 in [43], Theorem 3 in [45]) *Let* $\mathrm{lfs}_{\mathcal{M}}(p)$ *be the local feature size at* p *with respect to a mesh* $\mathcal{M}$ *(treating* $\mathcal{M}$ *as a PSLG), whereas* $\mathrm{lfs}(p)$ *remains the local feature size at* p *with respect to the input PSLG. Suppose a mesh* $\mathcal{M}$ *with smallest angle* θ *has the property that there is some constant* $k_1 \geq 1$*, such that for every point* p*,* $k_1 \mathrm{lfs}_{\mathcal{M}}(p) \geq \mathrm{lfs}(p)$*. Then the cardinality of* $\mathcal{M}$ *is less than* k_2 *times the cardinality of any other mesh of the input PSLG with smallest angle* θ*, where* $k_2 = \mathcal{O}\left(k_1^2/\theta\right)$*.*

Smaller values of δ offer more flexibility to a δ-graded Delaunay refinement algorithm in choosing Steiner points. However, from Lemma 5 it follows that as δ approaches $\sqrt{2}/\bar{\rho}$, C_T and C_S approach infinity, which leads to the worsening of the good grading guarantees. For example, as we can see from Figures 9, 10, and 11, the insertion of Steiner points at the boundaries of Type II selection disks increases the size of the Chesapeake bay mesh by about 0.5% over the circumcenter point insertion. If, furthermore, we drop the theoretical good grading guarantees and insert the points at the boundaries of Type I selection disks, the size of the Chesapeake bay mesh increases by about 15%. Therefore, along with satisfying application-specific requirements, the point insertion schemes should try to place Steiner points as close to circumcenters as possible.

3 Parallel Generalized Delaunay Refinement

In this section, we develop local Delaunay-independence conditions and show how quadtree leaves can be used to select subsets of candidate Steiner points for concurrent insertion. We extend our previous work [10, 13] by eliminating the use of the global circumradius upper bound and adapting the size of refinement and buffer zones to the user-defined grading function.

3.1 Delaunay-independent points

We expect our parallel Delaunay refinement algorithm to insert multiple Steiner points (one per selection disk) concurrently in such a way that it maintains the conformity and the Delaunay property of the mesh. Figure 12 illustrates how the concurrently inserted points can violate one of

these conditions. Edelsbrunner and Guoy [25] define two Steiner points as independent if the closures of their *prestars* (or *cavities* [27]) are disjoint. Their approach does not provide a way to avoid computing the cavities and their intersections for all candidate points, which is very expensive.

Definition 4 (Delaunay-independence) *Points p_i and p_j are* Delaunay-independent *with respect to mesh $\mathcal{M} = (V, T, S)$ if their concurrent insertion yields the conformal Delaunay mesh $\mathcal{M}' = (V \cup \{p_i, p_j\}, T', S')$. Otherwise, p_i and p_j are* Delaunay-conflicting.

Suppose point p_i encroaches upon a constrained segment s_i. Then p_i will not be inserted, and the midpoint p_i' of s_i will be inserted instead (similarly for p_j).

Definition 5 (Strong Delaunay-independence) *Points p_i and p_j are* strongly Delaunay-independent *with respect to mesh $\mathcal{M}$ iff any pair of points in $\{p_i, p_i'\} \times \{p_j, p_j'\}$ are Delaunay-independent with respect to $\mathcal{M}$.*

3.2 Local Delaunay-independence conditions

Lemma 6 (Delaunay-independence criterion) *Suppose the mesh $\mathcal{M} = (V, T, S)$ is conformal and Delaunay, and p_i and p_j are candidate Steiner points. Then p_i and p_j are Delaunay-independent iff*

$$\mathcal{C}_{\mathcal{M}}(p_i) \cap \mathcal{C}_{\mathcal{M}}(p_j) = \emptyset, \tag{16}$$

and

$$\forall e(p_m p_n) \in \partial\mathcal{C}_{\mathcal{M}}(p_i) \cap \partial\mathcal{C}_{\mathcal{M}}(p_j) \ : \ p_i \notin \bigcirc(\triangle p_j p_m p_n). \tag{17}$$

Proof First, $\mathcal{M}' = (V \cup \{p_i, p_j\}, T', S')$ is conformal iff (16) holds. Indeed, if (16) holds, then considering (1), the concurrent retriangulation of $\mathcal{C}_{\mathcal{M}}(p_i)$ and $\mathcal{C}_{\mathcal{M}}(p_j)$ will not yield overlapping triangles, and the mesh will be conformal. Conversely, if (16) does not hold, the newly created edges will intersect as shown in Fig. 12(left), and $\mathcal{M}'$ will not be conformal.

Now, we will show that $\mathcal{M}'$ is Delaunay iff (17) holds. The Delaunay Lemma [27] states that the triangulation is globally Delaunay if and only if the empty circumdisk criterion holds for every pair of adjacent triangles. Disregarding the symmetric cases, there are three types of pairs of adjacent triangles t_r and t_s, where $t_r \in \mathcal{B}_{\mathcal{M}'}(p_i)$, that will be affected: (i) $t_s \in \mathcal{B}_{\mathcal{M}'}(p_i)$, (ii)

$t_s \in T' \setminus \mathcal{B}_{\mathcal{M}'}(p_i) \setminus \mathcal{B}_{\mathcal{M}'}(p_j)$, and (iii) $t_s \in \mathcal{B}_{\mathcal{M}'}(p_j)$. The sequential Delaunay refinement algorithm guarantees that t_r and t_s will be locally Delaunay in the first two cases. In addition, condition (17) ensures that they will be locally Delaunay in the third case. Therefore, the mesh will be globally Delaunay. Conversely, if (17) does not hold, triangles $\triangle p_i p_m p_n$ and $\triangle p_j p_m p_n$ will not be locally Delaunay, and the mesh will not be globally Delaunay.

Corollary 1 (Sufficient condition of Delaunay-independence I [10]) *From Lemma 6 it follows that if* (16) *holds and* $\partial\mathcal{C}_{\mathcal{M}}(p_i) \cap \partial\mathcal{C}_{\mathcal{M}}(p_j) = \emptyset$*, then* p_i *and* p_j *are Delaunay-independent.*

To prove a more practical sufficient condition of Delaunay-independence we are going to use the following lemma which was proven in [10]:

Lemma 7 *Let* $\triangle p_k p_m p_n \in \mathcal{C}(p_j)$ *and* $\triangle p_l p_n p_m \notin \mathcal{C}(p_j)$*. Then*

$$r(\triangle p_j p_m p_n) < \max\{r(\triangle p_k p_m p_n), r(\triangle p_l p_n p_m)\}.$$

The following lemma is the main theoretical basis for the construction of the quadtree which will be described later.

Lemma 8 (Sufficient condition of Delaunay-independence II) *Points* p_i *and* p_j *are Delaunay-independent if there exists a subsegment* s *of segment* $\mathcal{L}(p_i p_j)$ *such that all triangle circumdisks which intersect* s *have diameter less than or equal to the length of* s*, i.e.,*

$$\exists s \subseteq \mathcal{L}(p_i p_j) \ : \ \forall t \in T \ : \ s \cap \bigcirc(t) \neq \emptyset \implies 2r(t) \leq |s|, \tag{18}$$

where $|s|$ *is the length of* s*.*

Proof First, condition (18) implies that $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$. Indeed, if there had been a triangle circumdisk which included both p_i and p_j, then the diameter of this circumdisk would be greater than the length of $\mathcal{L}(p_i p_j)$ which would contradict (18).

Now, there are two possibilities:

(i) If $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) = \emptyset$, then, by Corollary 1, p_i and p_j are Delaunay-independent.

(ii) Otherwise, let $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) \neq \emptyset$ and $e(p_m p_n)$ be an arbitrary edge in $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$. We are going to prove that $p_i \notin \bigcirc(\triangle p_j p_m p_n)$ and, thus, p_i and p_j are Delaunay-independent by Lemma 6. The proof is by contradiction. Suppose condition (18) holds and $p_i \in \bigcirc(\triangle p_j p_m p_n)$. There are two cases:

(ii-a) If $r(\triangle p_k p_m p_n) > r(\triangle p_l p_n p_m)$, see Figure 13, then from Lemma 7 it follows that

$$r(\triangle p_j p_m p_n) < r(\triangle p_k p_m p_n). \tag{19}$$

In addition, the assumption that $p_i \in \bigcirc(\triangle p_j p_m p_n)$ implies that

$$|\mathcal{L}(p_i p_j)| < 2r(\triangle p_j p_m p_n). \tag{20}$$

From (19) and (20) we conclude that the following relation holds:

$$|\mathcal{L}(p_i p_j)| < 2r(\triangle p_k p_m p_n). \tag{21}$$

Due to (21) and the assumption that (18) holds as well as the fact that $|s| \leq |\mathcal{L}(p_i p_j)|$, we conclude that s cannot intersect $\bigcirc(\triangle p_k p_m p_n)$. If p_r is the point of intersection of $\mathcal{L}(p_i p_j)$ with the boundary of $\bigcirc(\triangle p_k p_m p_n)$, then s is restricted to be the subsegment of $\mathcal{L}(p_i p_r)$ and

$$|s| \leq |\mathcal{L}(p_i p_r)|. \tag{22}$$

From the assumptions that $p_i \in \bigcirc(\triangle p_j p_m p_n)$ and $p_i \notin \bigcirc(\triangle p_k p_m p_n)$, it follows that p_i has to lie in the crescent-shaped area which is shaded in the figure and the following two relations hold:

$$s \cap \bigcirc(\triangle p_l p_n p_m) \neq \emptyset \tag{23}$$

and

$$|\mathcal{L}(p_i p_r)| < 2r(\triangle p_l p_n p_m). \tag{24}$$

Relations (22), (23), and (24) together imply that the condition (18) does not hold and we

have come to a contradiction.

(ii-b) If $r\left(\triangle p_k p_m p_n\right) \leq r\left(\triangle p_l p_n p_m\right)$, see Figure 14, then from Lemma 7 it follows that $r\left(\triangle p_j p_m p_n\right) < r\left(\triangle p_l p_n p_m\right)$ and considering that $|s| \leq |\mathcal{L}\left(p_i p_j\right)| < 2r\left(\triangle p_j p_m p_n\right) < 2r\left(\triangle p_l p_n p_m\right)$ we conclude that s cannot intersect $\bigcirc\left(\triangle p_l p_n p_m\right)$. This limits s to lie within the subsegment $\mathcal{L}\left(p_j p_r\right)$, where p_r is the point of intersection of $\mathcal{L}\left(p_i p_j\right)$ with the boundary of $\bigcirc\left(\triangle p_l p_n p_m\right)$; therefore,

$$|s| \leq |\mathcal{L}\left(p_j p_r\right)|. \tag{25}$$

The subsegment $\mathcal{L}\left(p_j p_r\right)$ lies completely inside the crescent-shaped region shaded in the figure which in turn is completely inside $\bigcirc\left(\triangle p_k p_m p_n\right)$, hence the following two relations hold:

$$s \cap \bigcirc\left(\triangle p_k p_m p_n\right) \neq \emptyset \tag{26}$$

and

$$|\mathcal{L}\left(p_j p_r\right)| < 2r\left(\triangle p_k p_m p_n\right). \tag{27}$$

Relations (25), (26), and (27) together imply that the condition (18) does not hold and we have come to a contradiction.

3.3 Quadtree construction

Callahan and Kosaraju [7, 8] developed a binary tree data structure for constructing well-separated pair decompositions of points, which was motivated by an application in n-body simulations [28]. They say that point sets A and B are *well-separated* if the rectangles which enclose A and B can each be contained in d-balls of radius r whose minimum distance is at least sr, where s is the *separation*. This data structure is based on a fair split tree of a point set which associates a leaf with each of the points. The construction of the quadtree which we describe below also uses a notion of separated regions. However, in the mesh generation context, the separation is based on the size and shape of the triangles in the underlying mesh. Another distinction is the introduction of the adjustable *granularity* parameter which allows to reduce the overheads associated with tree updates

by increasing the number of triangles per leaf. Finally, unlike in n-body simulations, in mesh refinement we have the creation of new points throughout the execution; therefore, the tree needs to be constructed dynamically.

Definition 6 (α-neighborhood) *Let the* α-neighborhood $\mathcal{N}_\alpha(L_i)$ ($\alpha \in \{Left, Right, Top, Bottom\}$) of quadtree leaf L_i *be the set of quadtree leaves that share a side with* L_i *and are located in the* α *direction of* L_i*. For example, in Fig. 15,* $L_k \in \mathcal{N}_{Top}(L_i)$ *and* $L_l \in \mathcal{N}_{Right}(L_i)$*.*

Definition 7 (Orthogonal directions) *Let the* orthogonal directions $\mathrm{ORT}(\alpha)$ *of direction* α *be*

$$\mathrm{ORT}(\alpha) = \begin{cases} \{Left, Right\} & \text{if } \alpha \in \{Top, Bottom\}, \\ \{Top, Bottom\} & \text{if } \alpha \in \{Left, Right\}. \end{cases}$$

Definition 8 (Buffer zone) *Let the set of leaves*

$$\mathrm{BUF}(Leaf) = \bigcup_{\alpha} \left(\mathcal{N}_\alpha(Leaf) \cup \bigcup_{L \in \mathcal{N}_\alpha(Leaf)} \bigcup_{\beta \in \mathrm{ORT}(\alpha)} \mathcal{N}_\beta(L) \right), \tag{28}$$

under the condition that the following relation holds

$$\forall L \in \mathrm{BUF}(Leaf), \forall t \in T \ : \ \bigcirc(t) \cap L \neq \emptyset \implies r(t) < \frac{1}{4}\ell(L), \tag{29}$$

where $\ell(L)$ *is the length of the side of* L*, be called a* buffer zone *of leaf Leaf with respect to mesh* $\mathcal{M}$*.*

Equation (29) is the criterion for the dynamic construction of the quadtree. Starting with the root node which covers the entire domain, each node of the quadtree is split into four smaller nodes as soon as all triangles, whose circumdisks intersect this node, have circumradii smaller than one eighth of its side length.

Definition 9 (Delaunay-separated regions) *Let two regions* $R_i \subset \mathbb{R}^2$ *and* $R_j \subset \mathbb{R}^2$ *be called* Delaunay-separated *with respect to mesh* $\mathcal{M}$ *iff any two arbitrary points* $p_i \in R_i$ *and* $p_j \in R_j$ *are strongly Delaunay-independent.*

Lemma 9 (Sufficient condition of leaf Delaunay-separateness) *If L_i and L_j are quadtree leaves, $i \neq j$, and $L_j \notin \text{BUF}(L_i)$, then L_i and L_j are Delaunay-separated.*

Proof First, for an arbitrary pair of points $p_i \in L_i$ and $p_j \in L_j \notin \text{BUF}(L_i)$, we will prove that p_i and p_j are Delaunay-independent. Then we will extend the proof to show that any pair of points from $\{p_i, p_i'\} \times \{p_j, p_j'\}$ are Delaunay-independent, which will imply that p_i and p_j are strongly Delaunay-independent; hence, L_i and L_j are Delaunay-separated.

By enumerating all possible configurations of leaves in $\text{BUF}(L_i)$ and grouping similar cases, without loss of generality all arrangements can be accounted for using the following argument.

Suppose $\mathcal{L}(p_i p_j)$ intersects the common boundary of L_i and L_k, where $L_k \in \mathcal{N}_{Top}(L_i) \subset \text{BUF}(L_i)$, see Figure 17. For each of the following sub-cases we show that there exists a subsegment s of segment $\mathcal{L}(p_i p_j)$ which satisfies the condition of Lemma 8, and therefore p_i and p_j are Delaunay-independent:

(i) If $\mathcal{L}(p_i p_j)$ intersects the upper boundary of L_k, see Figure 17(left), then we choose s as the intersection of $\mathcal{L}(p_i p_j)$ with L_k and note that $|s| \geq \ell(L_k)$ while all triangle circumdisks which intersect L_k have diameter less than $\ell(L_k)$.

(ii) Otherwise, let $\mathcal{L}(p_i p_j)$ intersect the left boundary of L_k and let $L_m \in \mathcal{N}_{Left}(L_k) \subset \text{BUF}(L_i)$ be the leaf adjacent to this boundary at the point of intersection. There are two sub-cases:

(ii-a) If $\mathcal{L}(p_i p_j)$ intersects the upper boundary of L_m, see Figure 17 (center), then we select s at the intersection of $\mathcal{L}(p_i p_j)$ with $L_k \cup L_m$. In this case, $|s| \geq \ell(L_k)$ and all triangle circumdisks which intersect s have diameter less than $\max\{\ell(L_k), \ell(L_m)\} = \ell(L_k)$.

(ii-b) If $\mathcal{L}(p_i p_j)$ intersects the left boundary of L_m, see Figure 17(right), then we select s at the intersection of $\mathcal{L}(p_i p_j)$ with L_m. In this case, $|s| \geq \ell(L_m)$ and all triangle circumdisks which intersect s have diameter less than $\ell(L_m)$.

Now, suppose p_i and p_j encroach upon constrained edges $e(p_l p_m)$ and $e(p_r p_s)$, respectively (Fig. 16). Then the midpoints p_i' and p_j' of $e(p_l p_m)$ and $e(p_r p_s)$ will be inserted instead. If p_i' and p_j' lie in the same quadtree leaves as p_i and p_j, then they can be proven Delaunay-independent using the argument above.

Let us analyze the worst case, i.e., $p'_i, p'_j \in L_k \in \mathrm{BUF}\,(L_i)$. Since the diametral disk of an edge has the smallest radius among all disks whose circle passes through the endpoints of the edge, then $r\,(e\,(p_l p_m)) \leq r\,(\triangle p_l p_m p_n) < \frac{1}{4}\ell\,(L_k)$ and $r\,(e\,(p_r p_s)) \leq r\,(\triangle p_r p_s p_t) < \frac{1}{4}\ell\,(L_k)$. Therefore, $\|p'_i - p'_j\| > \frac{1}{2}\ell\,(L_k)$. We can construct imaginary buffer squares L_{k_1} and L_{k_2} between p'_i and p'_j inside leaf L_k with $\ell\,(L_{k_1}) = \ell\,(L_{k_2}) = \frac{1}{2}\ell\,(L_k)$. Since by construction all triangle circumdisks which intersect L_k have radius less than $\frac{1}{4}\ell\,(L_k)$, then all triangle circumdisks which intersect L_{k_1} or L_{k_2} will have radius less than $\frac{1}{2}\ell\,(L_{k_1})$ or $\frac{1}{2}\ell\,(L_{k_2})$. Then, by an argument similar to the one above, we show that for p'_i and p'_j the condition of Lemma 8 is satisfied, and hence these two points are Delaunay-independent.

3.4 Parallel algorithm

If a part of the mesh associated with a leaf *Leaf* of the quadtree is scheduled for refinement by a thread, no other thread can refine the parts of the mesh associated with the buffer zone $\mathrm{BUF}\,(\mathit{Leaf})$ of this leaf. To simplify the presentation, here we rewrite the definition of the buffer zone in the way it is used by the algorithm. For this purpose, we introduce a superscript to the $\mathrm{BUF}\,(\cdot)$ symbol:

$$\begin{aligned} &\mathrm{BUF}^1\,(\mathit{Leaf}) = \mathrm{BUF}\,(\mathit{Leaf})\,, \\ &\mathrm{BUF}^i\,(\mathit{Leaf}) = \mathrm{BUF}^{i-1}\,(\mathit{Leaf}) \cup \bigcup_{L \in \mathrm{BUF}^{i-1}(\mathit{Leaf})} \mathrm{BUF}\,(L)\,, \quad i \geq 2. \end{aligned} \tag{30}$$

The algorithm is designed for the execution by one master thread which manages the work pool and multiple refinement threads which refine the mesh and the quadtree. Figure 18 presents a high level description of the Parallel Generalized Delaunay Refinement (PGDR) algorithm performed by the master thread. Line 14 shows the invocation of a refinement thread from the master thread. Figure 19 presents the part of the algorithm executed by each of the refinement threads.

When a quadtree leaf *Leaf* is scheduled for refinement, we remove not just $\mathrm{BUF}^1\,(\mathit{Leaf})$ but $\mathrm{BUF}^2\,(\mathit{Leaf})$ from the refinement queue. Although this is not required by our theory, there are two implementation considerations for doing so, and both are related to the goal of reducing fine-grain synchronization.[1] First, each leaf has an associated data structure which stores the poor quality

[1] As we have shown previously [1], the overheads of portable thread packages (e.g., Pthreads) on modern SMTs are

triangles whose circumdisks intersect this leaf, so that we can maintain the relation (29). Therefore, we would have to introduce synchronization in line 9 of the algorithm in Figure 19 to maintain this data structure. Second, for efficiency considerations, we followed the design of the triangle data element that is used in `Triangle` [46]. In particular, each triangle contains pointers to neighboring triangles for easy mesh traversal. However, if two cavities share an edge and are updated by the concurrent threads, which can be done legitimately in certain cases, these triangle–neighbor pointers will be invalidated. For these reasons, we chose to completely separate the sets of leaves affected by the mesh refinement performed by multiple threads.

Each of the worker threads performs the refinement of the mesh and the refinement of the quadtree. The poor quality triangles whose split-points selected by a deterministic function $f(\cdot)$ are inside the square of *Leaf* are stored in the data structure denoted here as $PoorTriangles(Leaf)$. *Leaf* needs to be scheduled for refinement if the size of this data structure is not empty. In addition, each *Leaf* has a counter for the triangles with various ratios of the side length of *Leaf* to their circumradius. If we denote $\sigma(t, Leaf) = \left\lfloor \log_2 \frac{\ell(Leaf)}{r(t)} \right\rfloor$, then
$Counter_i(Leaf) = |\{t \in \mathcal{M} \mid (\bigcirc(t) \cap Leaf \neq \emptyset) \wedge (\sigma(t, Leaf) = i)\}|$.
When $Counter_i(Leaf) = 0$, $\forall i = 1, 2, 3$, it implies that (29) would hold for each of the children of *Leaf*, and *Leaf* can be split. Lemma 7 guarantees that when a point is inserted into a Delaunay mesh using the B-W algorithm, the circumradii of the new triangles are not going to be larger than the circumradii of the triangles in the cavity of the point or those that are adjacent to the cavity. In addition, the following lemma proves that the circumdisks of the new triangles are not going to extend beyond the circumdisks of the triangles in the cavity and the triangles adjacent to the cavity. Therefore, new triangles that would violate (29) are not going to be created.

Lemma 10 *Let* $\triangle p_k p_m p_n \in \mathcal{C}(p_j)$ *and* $\triangle p_l p_n p_m \notin \mathcal{C}(p_j)$. *Then*

$$\bigcirc(\triangle p_j p_m p_n) \subset \Big(\bigcirc(\triangle p_k p_m p_n) \cup \bigcirc(\triangle p_l p_n p_m) \Big).$$

Proof Consider Figure 20. Let p_4 be the midpoint of edge $e(p_m p_n)$, and let the following points lie at the intersections of the perpendicular bisector of $e(p_m p_n)$ with the boundaries of the

not small enough to tolerate fine-grain parallelism in Delaunay mesh refinement.

corresponding circumdisks: p_1 and p_5 with $\bigcirc(\triangle p_l p_n p_m)$, p_2 and p_6 with $\bigcirc(\triangle p_j p_m p_n)$, and p_3 and p_7 with $\bigcirc(\triangle p_k p_m p_n)$. Due to the premise that $p_j \in \bigcirc(\triangle p_k p_m p_n)$ and $p_j \notin \bigcirc(\triangle p_l p_n p_m)$, p_6 is restricted to lie between p_5 and p_7, and p_2 is restricted to lie between p_1 and p_3. Therefore, the arc $p_n p_6 p_j p_m$ is restricted to lie within the shaded region which is $\bigcirc(\triangle p_k p_m p_n) \setminus \bigcirc(\triangle p_l p_n p_m)$ and the arc $p_m p_2 p_n$ is restricted to lie within the shaded region which is $\bigcirc(\triangle p_l p_n p_m) \setminus \bigcirc(\triangle p_k p_m p_n)$. Hence, $\bigcirc(\triangle p_j p_m p_n)$ cannot extend beyond $\bigcirc(\triangle p_k p_m p_n) \cup \bigcirc(\triangle p_l p_n p_m)$.

Each leaf of the quadtree has associated with it a bucketing structure which holds poor quality triangles:

$$PoorTriangles_i(Leaf) = \{t \in \mathcal{M} \mid (f(t) \in Leaf) \wedge (\sigma(t, Leaf) = i) \wedge ((\mathcal{A}(t) > \bar{\mathcal{A}}(t)) \vee (\rho(t) > \bar{\rho}))\}.$$

At each mesh refinement step, all triangles in $PoorTriangles_j(Leaf)$ are refined, for all $j = i_{min}, \ldots, i_{min} + granularity$, where $i_{min} = \min_{PoorTriangles_i(Leaf) \neq \emptyset} i$, and $granularity \geq 1$ is a parameter that controls how much computation is done during a single mesh refinement call. After a mesh refinement call returns, the feasibility of splitting *Leaf* is evaluated, and it is recursively subdivided if necessary.

The $PoorTriangles$ structure allows our parallel algorithm to give priority to triangles with large circumradii. As discussed in [45, 47], Ruppert's sequential Delaunay refinement algorithm has quadratic worst-case running time, even though in most practical cases the time is linear with respect to the output size. Recently, Miller [41] proposed a Delaunay refinement algorithm which runs in optimal $\mathcal{O}(n \log n + m)$ time, where n is the size of the input, and m is the size of the output. He achieved this improvement by introducing a priority queue, where the skinny triangles are ordered by their diameter (equivalently, circumradius), and the triangles with the largest diameter are refined first. Although our algorithm does not introduce total ordering as Miller's sequential algorithm, it allows to eliminate quadratic running time for pathological input geometries. Spielman, Teng, and Ünggör [48, 49] presented the first theoretical analysis of the complexity of parallel Delaunay refinement algorithms.

3.5 Implementation and evaluation

We developed two implementations of the PGDR algorithm. The first one is written in Python using Python threads module, and the second one is in C++ using Pthreads. The Python code is interpreted and, thus, is much slower than the compiled code written in languages like C/C++, however, it offers high level data types and expressions which allow to significantly decrease the development cycle. We ran this code on a Linux box with two single-core Pentium-4 processors. Figures 21 and 22 compare the meshes produced by our Python implementation and `Triangle` library [46] for a pipe cross-section and a key. Figure 23 also shows the initial geometry and the quadtree produced by our algorithm for the cylinder flow problem [23]. For all of the quadtree nodes, mesh refinement and node subdivision routines were applied concurrently while preserving the required buffer zones, until the quality constraints were met. The specified grading functions were used as follows. If (x_i, y_i) is the centroid of the triangle t_i, then the area of t_i has to be less than $\bar{\mathcal{A}}(x_i, y_i)$. In all experiments we used the same minimal angle bound of 20°. These tests indicate that while maintaining the required quality of the elements, the number of triangles produced by our method is close, and sometimes is even smaller, than produced by `Triangle` [46].

The experiments with the code written in C++ were conducted on an IBM Power5 node with two dual-core processors running at 1.6 GHz and 8 GBytes of total physical memory. We compared our implementation with the fastest to our knowledge sequential Delaunay mesh generator `Triangle` version 1.6 [46]. This is the latest release of `Triangle`, which uses the off-center point insertion algorithm [50]. In order to make the results comparable, our PGDR implementation also uses the off-center point insertion [50]. `Triangle` provides a convenient facility for the generation of meshes respecting user-defined area bounds. The user can write his own `triunsuitable()` function and link it against `Triangle`. This function is called to examine each new triangle and to decide whether or not it should be considered big and enqueued for refinement. We encoded our grading function into the `triunsuitable()` function, compiled it into an object file, and linked against both `Triangle` and our own PGDR implementation. We ran each of the tests 10 times and used average or median timing measurements as indicated.

Figure 24(left) presents the total running times for several granularity values, as the number of compute threads increases from 1 to 4. One additional thread was used to manage the refinement

queue. The mesh was constructed for the pipe cross-section model shown in Figure 6(right), using the grading function

$$\bar{\mathcal{A}}\,(x, y) = 10^{-4}(\sqrt{(x-200)^2 + (y-200)^2} + 1).$$

The total number of triangles produced both by `Triangle` and PGDR was approximately 17 million.

We can see that the best running time was achieved using 4 compute threads with the granularity value equal to 2, and it amounted for 56% of `Triangle`'s sequential running time. It is also interesting to see the intersection of lines corresponding to granularities 2 and 3, when the number of compute threads was increased from 3 to 4. This intersection reflects one of the basic tradeoffs in parallel computing, between granularity and concurrency: in order to increase the concurrency we have to decrease the granularity, which introduces more overheads.

Figure 24(right) shows the breakdown of the total execution time for each of the threads. The fact that the management thread is idle for 93% of the total time suggests the possibility of high scalability of the code on larger machines, since it can handle many more refinement threads (cores) than the current widely available machines have.

Standard system memory allocators exhibited high latency and poor scalability in our experiments, which lead us to develop a custom memory management class. At initialization, our memory pool class takes the size of the underlying object (triangle, vertex, quadtree node, or quadtree junction point) as a parameter and at runtime it allocates blocks of memory which can fit a large number of objects. When the objects are deleted, they are not deallocated but are kept for later reuse instead. Each thread manages a separate set of memory pools, which allows us to avoid synchronization. Our quantitative study of the performance of the standard, the custom, and a novel generic multiprocessor allocator appears elsewhere [9].

4 Conclusions

We analyzed the existing point insertion methods for guaranteed quality Delaunay refinement and unified them into a framework which allows to develop customized mesh optimization techniques. The goals of these techniques may include the following:

- minimizing the number of inserted points, see for example [50] and Subsection 2.2.2 here;

- eliminating slivers, see [18, 37, 38];
- splitting multiple poor quality triangles simultaneously, see Fig. 25(left).
- creating elongated edges in required directions, see Fig. 25(center);
- inserting more than one point, e.g., to create elements with specific shapes, see Fig. 25(right);
- satisfying other application-specific requirements, for example, conformity to a boundary zone, see Fig. 1.

We conducted experiments with three different point placement methods: circumcenter, off-center and a new optimization-based method which allows to improve the size of the mesh by up to 20% and up to 5% over the first two methods, respectively.

An extension of the selection disks to anisotropic mesh generation requires additional analysis. Labelle and Shewchuk [36] presented an anisotropic guaranteed-quality mesh generation algorithm. With each point p in Ω they associate a symmetric positive definite metric tensor which specifies how distances and angles are measured from the perspective of p. As a result, the Voronoi diagram of a point set becomes very complicated, and may even contain disconnected faces; therefore, it does not always dualize to a correct triangulation. The point insertion scheme developed in [36] takes into account the visibility of points with respect to Voronoi faces, which would also restrict the shape of a selection region.

We presented a theoretical framework for developing parallel Delaunay meshing codes, which allows to control the size of the elements with a user-defined grading function. We eliminated such disadvantages of the previously proposed methods as the necessity to maintain a cavity (conflict) graph, the rollbacks, the requirement to solve a difficult domain decomposition problem, and the centralized sequential resolution of potential conflicts. Our theory leverages the quality guarantees of the existing sequential Delaunay refinement algorithms. The experimental results confirm that the parallel algorithm produces meshes with the same quality as the sequential Delaunay refinement algorithm and does not lead to over-refinement.

We presented the algorithm and the implementation of a parallel 2D graded guaranteed quality Delaunay mesh generator. Our algorithm is designed to work with custom point placement

techniques which choose points from the selection disks. Our current algorithm is limited to deterministic point selection; incorporating randomized point selection is left to the future research. The experimental results show that our code on a machine with two dual-core processors runs in 56% of the time taken by the fastest sequential code `Triangle` [46]. By using a quadtree constructed in a specific way, we eliminated the need to solve the difficult domain decomposition problem. Our implementation is designed for shared memory architectures. In [10, 13] we described a parallel algorithm for distributed memory machines that allows to produce uniform meshes. In the uniform case, a simple static data distribution works very well. However, for non-uniform and adaptive meshes a dynamic work and/or data distribution is required. This problem can be addressed by the use of run-time load balancing approaches, see [3] for more details. Our ongoing research includes the extension of the theory and of the implementation to three dimensions.

5 Acknowledgments

We thank Gary Miller for helpful references. This work was supported (in part) by the NSF grants CCS-0750901, CCF-0833081, CCF-0916526, and by the John Simon Guggenheim Foundation.

References

[1] Christos D. Antonopoulos, Xiaoning Ding, Andrey N. Chernikov, Filip Blagojevic, Dimitris S. Nikolopoulos, and Nikos P. Chrisochoides. Multigrain parallel Delaunay mesh generation: Challenges and opportunities for multithreaded architectures. In *Proceedings of the 19th Annual International Conference on Supercomputing*, pages 367–376, Cambridge, MA, 2005. ACM Press.

[2] C. Armstrong, D. Robinson, R. McKeag, T. Li, S. Bridgett, R. Donaghy, and C. McGleenan. Medials for meshing and more. In *Proceedings of 4th International Meshing Roundtable*, pages 277–288. Sandia National Laboratories, 1995.

[3] Kevin Barker, Andrey Chernikov, Nikos Chrisochoides, and Keshav Pingali. A load balancing framework for adaptive and asynchronous applications. *IEEE Transactions on Parallel and Distributed Systems*, 15(2):183–192, February 2004.

[4] G. E. Blelloch, J.C. Hardwick, G. L. Miller, and D. Talmor. Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica*, 24:243–269, 1999.

[5] G. E. Blelloch, G. L. Miller, and D. Talmor. Developing a practical projection-based parallel Delaunay algorithm. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 186–195, Philadelphia, PA, May 1996.

[6] Adrian Bowyer. Computing Dirichlet tesselations. *Computer Journal*, 24:162–166, 1981.

[7] Paul B. Callahan and S. Rao Kosaraju. Algorithms for dynamic closest pair and n-body potential fields. In *Proceedings of the 6th annual ACM-SIAM symposium on Discrete algorithms*, pages 263–272, San Francisco, CA, 1995.

[8] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, 1995.

[9] Andrey Chernikov, Christos Antonopoulos, Nikos Chrisochoides, Scott Schneider, and Dimitris Nikolopoulos. Experience with memory allocators for parallel mesh generation on multicore architectures. In *10th International Conference on Numerical Grid Generation in Computational Field Simulations*, Forth, Crete, Greece, September 2007. Published on CD-ROM.

[10] Andrey N. Chernikov and Nikos P. Chrisochoides. Practical and efficient point insertion scheduling method for parallel guaranteed quality Delaunay refinement. In *Proceedings of the 18th Annual International Conference on Supercomputing*, pages 48–57, Malo, France, 2004. ACM Press.

[11] Andrey N. Chernikov and Nikos P. Chrisochoides. Parallel 2D graded guaranteed quality Delaunay mesh refinement. In *Proceedings of the 14th International Meshing Roundtable*, pages 505–517, San Diego, CA, September 2005. Springer.

[12] Andrey N. Chernikov and Nikos P. Chrisochoides. Generalized Delaunay mesh refinement: From scalar to parallel. In *Proceedings of the 15th International Meshing Roundtable*, pages 563–580, Birmingham, AL, September 2006. Springer.

[13] Andrey N. Chernikov and Nikos P. Chrisochoides. Parallel guaranteed quality Delaunay uniform mesh refinement. *SIAM Journal on Scientific Computing*, 28:1907–1926, 2006.

[14] Andrey N. Chernikov and Nikos P. Chrisochoides. Three-dimensional semi-generalized point placement method for Delaunay mesh refinement. In *Proceedings of the 16th International Meshing Roundtable*, pages 25–44, Seattle, WA, October 2007. Springer.

[15] Andrey N. Chernikov and Nikos P. Chrisochoides. Three-dimensional Delaunay refinement for multi-core processors. In *Proceedings of the 22nd Annual International Conference on Supercomputing*, pages 214–224, Island of Kos, Greece, 2008. ACM Press.

[16] L. Paul Chew. Guaranteed-quality triangular meshes. Technical Report TR89983, Cornell University, Computer Science Department, 1989.

[17] L. Paul Chew. Guaranteed quality mesh generation for curved surfaces. In *Proceedings of the 9th ACM Symposium on Computational Geometry*, pages 274–280, San Diego, CA, 1993.

[18] L. Paul Chew. Guaranteed-quality Delaunay meshing in 3D. In *Proceedings of the 13th ACM Symposium on Computational Geometry*, pages 391–393, Nice, France, 1997.

[19] Nikos Chrisochoides and Démian Nave. Parallel Delaunay mesh generation kernel. *International Journal for Numerical Methods in Engineering*, 58:161–176, 2003.

[20] Nikos P. Chrisochoides. A survey of parallel mesh generation methods. Technical Report BrownSC-2005-09, Brown University, 2005. Also appears as a chapter in Numerical Solution of Partial Differential Equations on Parallel Computers (eds. Are Magnus Bruaset and Aslak Tveito), Springer, 2006.

[21] Tim Culver. *Computing the Medial Axis of a Polyhedron Reliably and Efficiently.* PhD thesis, The University of North Carolina at Chapel Hill, 2000.

[22] Boris N. Delaunay. Sur la sphere vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Mataematicheskii i Estestvennyka Nauk*, 7:793–800, 1934.

[23] Suchuan Dong, Didier Lucor, and George Em Karniadakis. Flow past a stationary and moving cylinder: DNS at Re=10,000. In *Proceedings of the 2004 Users Group Conference (DOD_UGC'04)*, pages 88–95, Williamsburg, VA, 2004.

[24] Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*. Cambridge University Press, England, 2001.

[25] Herbert Edelsbrunner and Damrong Guoy. Sink-insertion for mesh improvement. In *Proceedings of the 17th ACM Symposium on Computational Geometry*, pages 115–123, Medford, MA, 2001.

[26] William H. Frey. Selective refinement: A new strategy for automatic node placement in graded triangular meshes. *International Journal for Numerical Methods in Engineering*, 24(11):2183–2200, 1987.

[27] Paul-Louis George and Houman Borouchaki. *Delaunay Triangulation and Meshing. Application to Finite Elements*. HERMES, 1998.

[28] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *Journal of Computational Physics*, 73(2):325–348, 1987.

[29] Halit Nebi Gürsoy. *Shape interrogation by medial axis transform for automated analysis*. PhD thesis, Massachusetts Institute of Technology, 1989.

[30] Sariel Har-Peled and Alper Üngör. A time-optimal delaunay refinement algorithm in two dimensions. In *Proceedings of the 21st annual symposium on Computational geometry*, pages 228–236, Pisa, Italy, 2005. ACM Press.

[31] Martin Isenburg, Yuanxin Liu, Jonathan Shewchuk, and Jack Snoeyink. Streaming computation of Delaunay triangulations. *ACM Transactions on Graphics*, 25(3):1049–1056, 2006.

[32] Clemens Kadow. Adaptive dynamic projection-based partitioning for parallel Delaunay mesh generation algorithms. In *SIAM Workshop on Combinatorial Scientific Computing*, San-Francisco, CA, February 2004.

[33] Clemens Kadow. *Parallel Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, 2004.

[34] Clemens Kadow and Noel Walkington. Design of a projection-based parallel Delaunay mesh generation and refinement algorithm. In *4th Symposium on Trends in Unstructured Mesh Generation*, Albuquerque, NM, July 2003.
http://www.andrew.cmu.edu/user/sowen/usnccm03/agenda.html.

[35] G.E. Karniadakis and S.A. Orszag. Nodes, modes, and flow codes. *Physics Today*, 46:34–42, 1993.

[36] Francois Labelle and Jonathan Richard Shewchuk. Anisotropic Voronoi diagrams and guaranteed-quality anisotropic mesh generation. In *Proceedings of the 19th ACM Symposium on Computational geometry*, pages 191–200, San Diego, CA, 2003.

[37] Xiang-Yang Li. Generating well-shaped d-dimensional Delaunay meshes. *Theoretical Computer Science*, 296(1):145–165, 2003.

[38] Xiang-Yang Li and Shang-Hua Teng. Generating well-shaped Delaunay meshes in 3D. In *Proceedings of the 12th annual ACM-SIAM symposium on Discrete algorithms*, pages 28–37, Washington, D.C., 2001.

[39] Leonidas Linardakis and Nikos Chrisochoides. Delaunay decoupling method for parallel guaranteed quality planar mesh refinement. *SIAM Journal on Scientific Computing*, 27(4):1394–1423, 2006.

[40] Leonidas Linardakis and Nikos Chrisochoides. Algorithm 870: A static geometric medial axis domain decomposition in 2D Euclidean space. *ACM Transactions on Mathematical Software*, 34(1):1–28, 2008.

[41] Gary L. Miller. A time efficient Delaunay refinement algorithm. In *Proceedings of the 15th annual ACM-SIAM symposium on Discrete algorithms*, pages 400–409, New Orleans, LA, 2004.

[42] Gary L. Miller, Dafna Talmor, Shang-Hua Teng, and Noel Walkington. A Delaunay based numerical method for three dimensions: Generation, formulation, and partition. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing*, pages 683–692, Las Vegas, NV, May 1995.

[43] Scott A. Mitchell. Cardinality bounds for triangulations with bounded minimum angle. In *Proceedings of the 6th Canadian Conference on Computational Geometry*, pages 326–331, Saskatoon, Saskatchewan, Canada, August 1994.

[44] Démian Nave, Nikos Chrisochoides, and L. Paul Chew. Guaranteed–quality parallel Delaunay refinement for restricted polyhedral domains. *Computational Geometry: Theory and Applications*, 28:191–215, 2004.

[45] Jim Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.

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

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

[48] Daniel A. Spielman, Shang-Hua Teng, and Alper Üngör. Parallel Delaunay refinement: Algorithms and analyses. In *Proceedings of the 11th International Meshing Roundtable*, pages 205–217, Ithaca, NY, 2001.

[49] Daniel A. Spielman, Shang-Hua Teng, and Alper Üngör. Time complexity of practical parallel Steiner point insertion algorithms. In *Proceedings of the 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 267–268, Barcelona, Spain, 2004. ACM Press.

[50] Alper Üngör. Off-centers: A new type of Steiner points for computing size-optimal guaranteed-quality Delaunay triangulations. In *Proceedings of LATIN*, pages 152–161, Buenos Aires, Argentina, April 2004.

[51] Roy A. Walters. Coastal ocean models: Two useful finite element methods. *Recent Developments in Physical Oceanographic Modeling: Part II*, 25:775–793, 2005.

[52] David F. Watson. Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes. *Computer Journal*, 24:167–172, 1981.

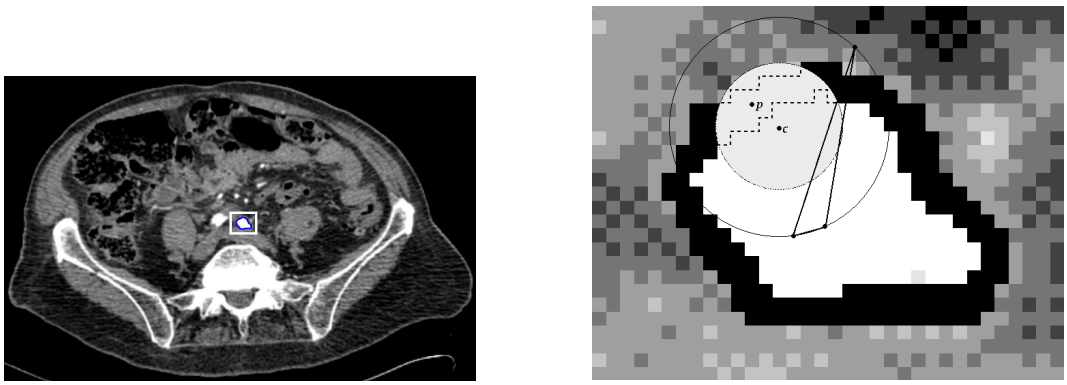

Figure 1: **(Left)** An MRI scan showing a cross-section of a body. **(Right)** A zoom-in of the selected area containing an artery: the inside is white, the outside has different shades of gray and the black zone is an approximate boundary between these regions. The standard Delaunay refinement algorithm would insert the circumcenter c. However, in order to construct a mesh which conforms to the boundary, another point (p) would be a better choice.

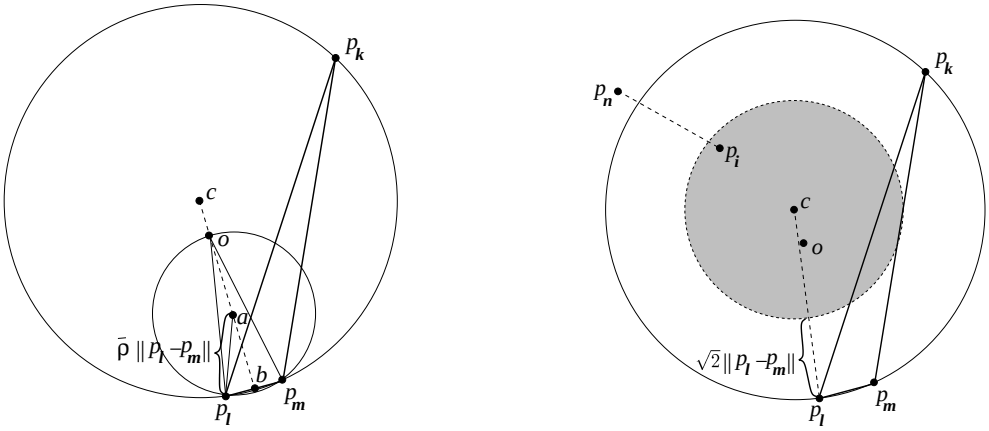

Figure 2: **(Left)** Delaunay refinement with the off-centers. [50] **(Right)** The Type I selection disk (shaded) for the insertion of a Steiner point.

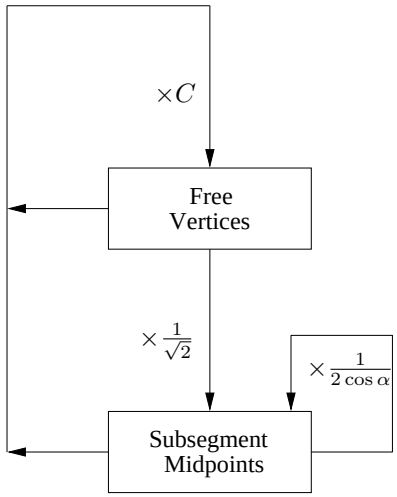

Figure 3: Flow diagram from [47] illustrating the relationship between the insertion radius of a vertex and its parent in two dimensions. If no cycle has a product smaller then one, the algorithm will terminate. Input vertices are not shown since they do not participate in cycles. In [47] the constant $C = \bar{\rho} \geq \sqrt{2}$. In our case, with the use of Type I selection disks $C = \sqrt{2}$, and with the use of Type II selection disks $C = \delta\bar{\rho} \geq \sqrt{2}$.

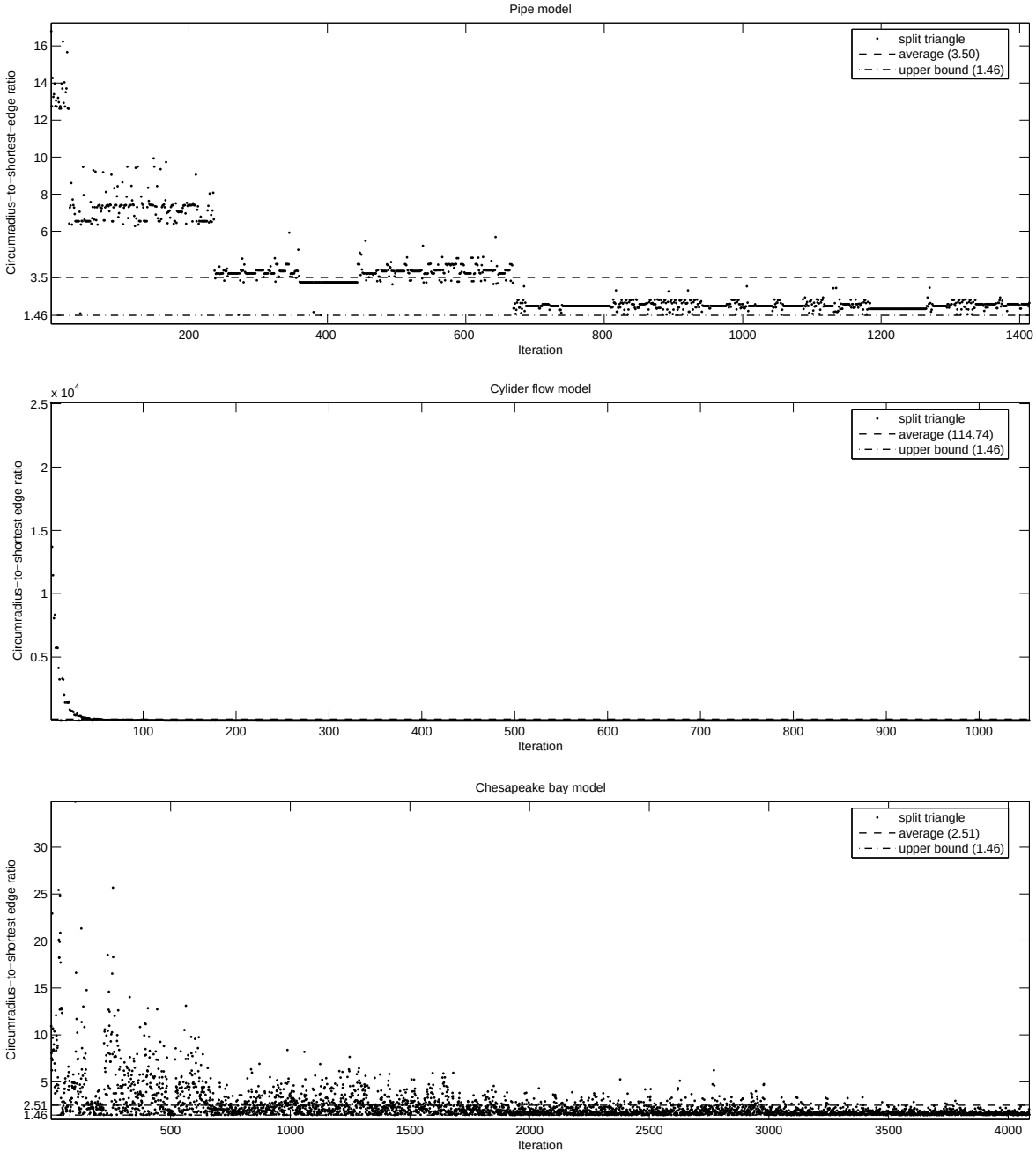

Figure 4: Circumradius-to-shortest edge length ratios of skinny triangles being eliminated by Delaunay refinement.

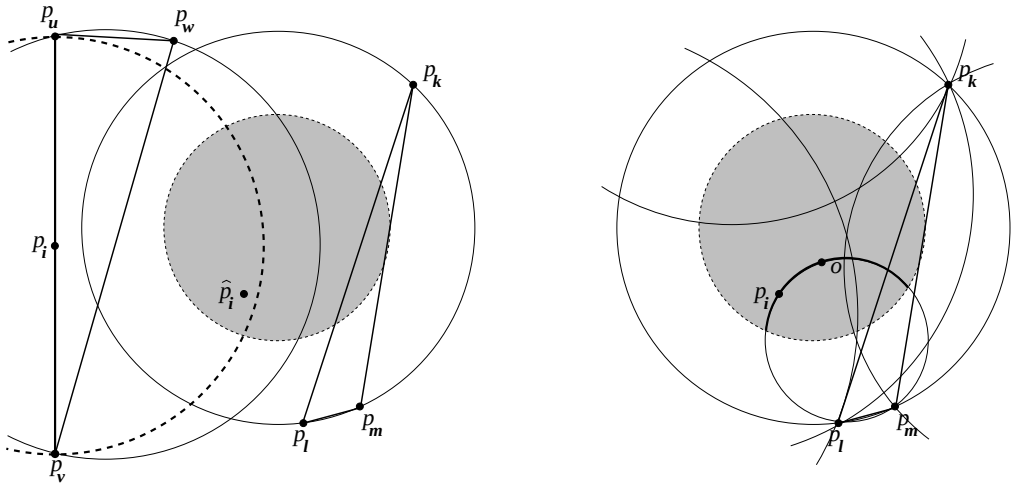

Figure 5: **(Left)** $\hat{p}_i$ is a Steiner point within a selection disk of a poor quality triangle which encroaches upon a constrained segment $e\,(p_u p_v)$. **(Right)** An example of an optimization-based method for the selection of a Steiner point within a selection disk of a poor quality triangle.

Figure 6: **(Left)** An upper part of a model of cylinder flow. **(Right)** Pipe cross-section model.

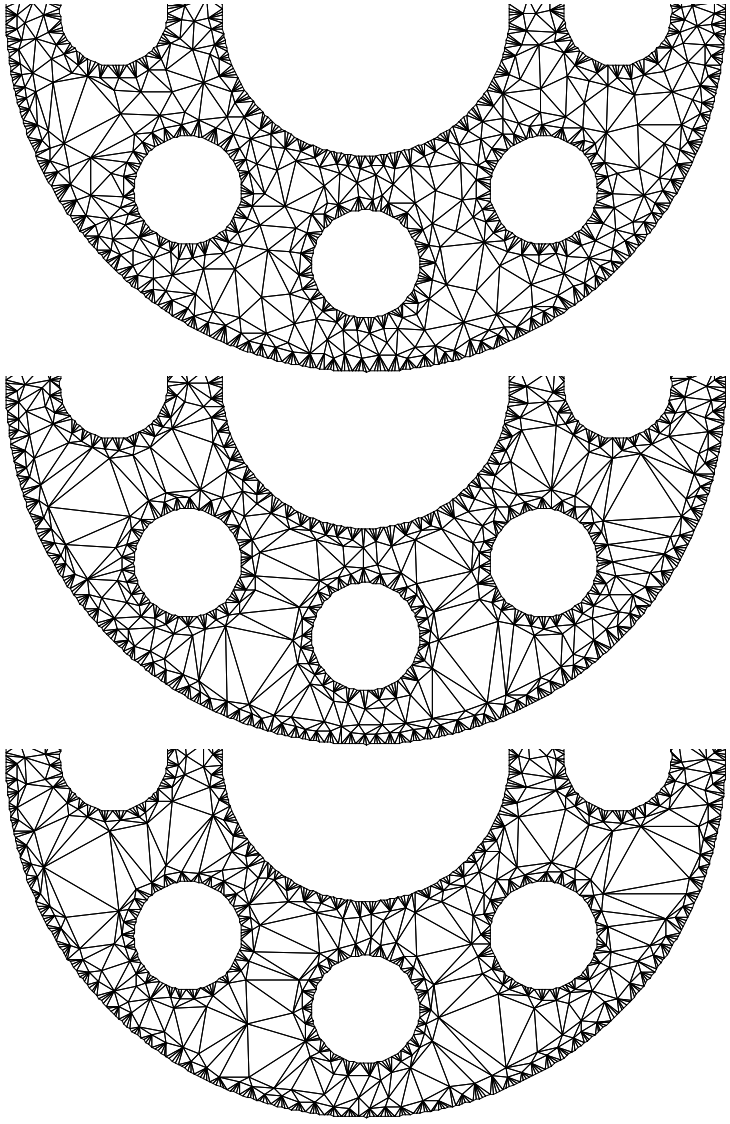

Figure 7: The pipe mesh (lower half is shown) with the minimal angle bound equal to 10°. **(Top)** Steiner points are inserted at the circumcenters of skinny triangles: 3033 triangles. **(Center)** Steiner points are inserted at the off-centers: 2941 triangles. **(Bottom)** Our optimization-based method: 2841 triangles.

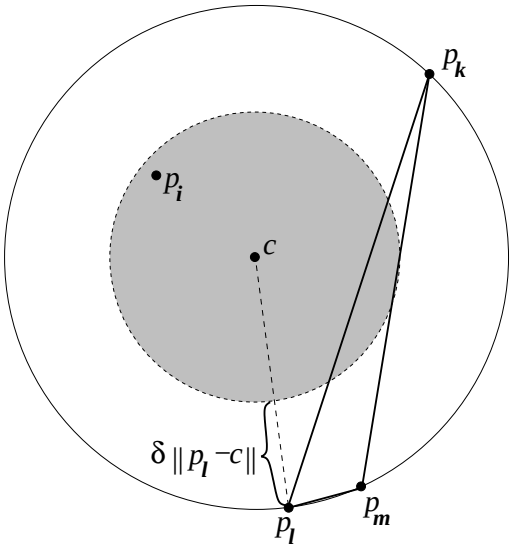

Figure 8: Selection of a Steiner point by a δ-graded Delaunay refinement algorithm. When $\delta = \sqrt{2}/\bar{\rho}$ the shaded disk corresponds to the Type II selection disk.

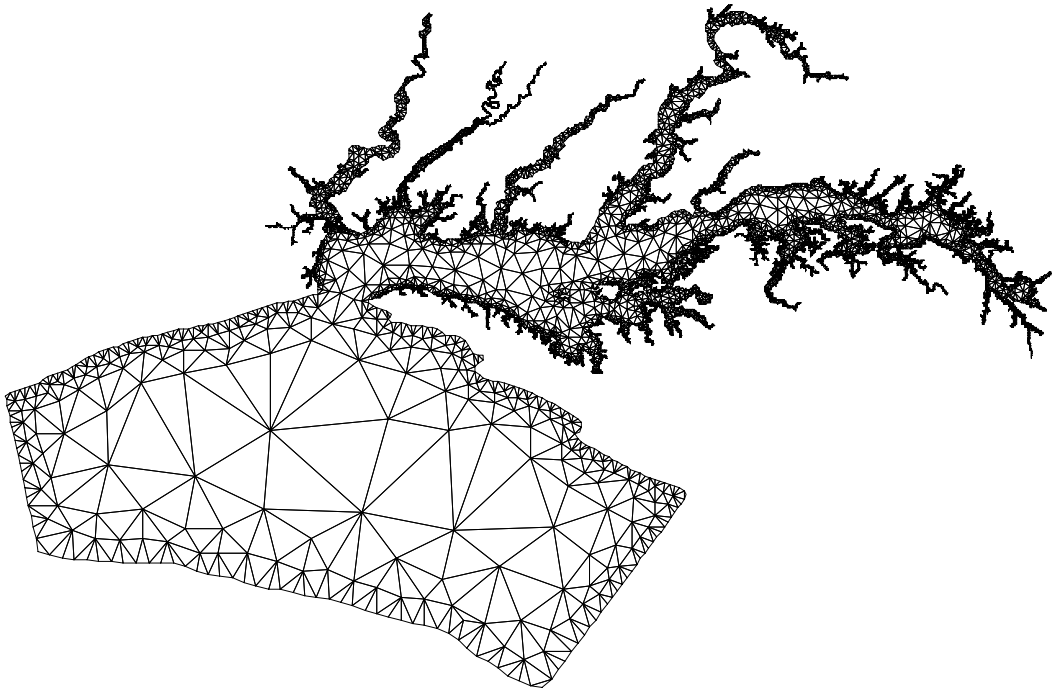

Figure 9: The Chesapeake bay mesh with the minimal angle bound equal to 20°. Steiner points are inserted at the circumcenter (c_x, c_y) of each skinny triangle: 22438 triangles.

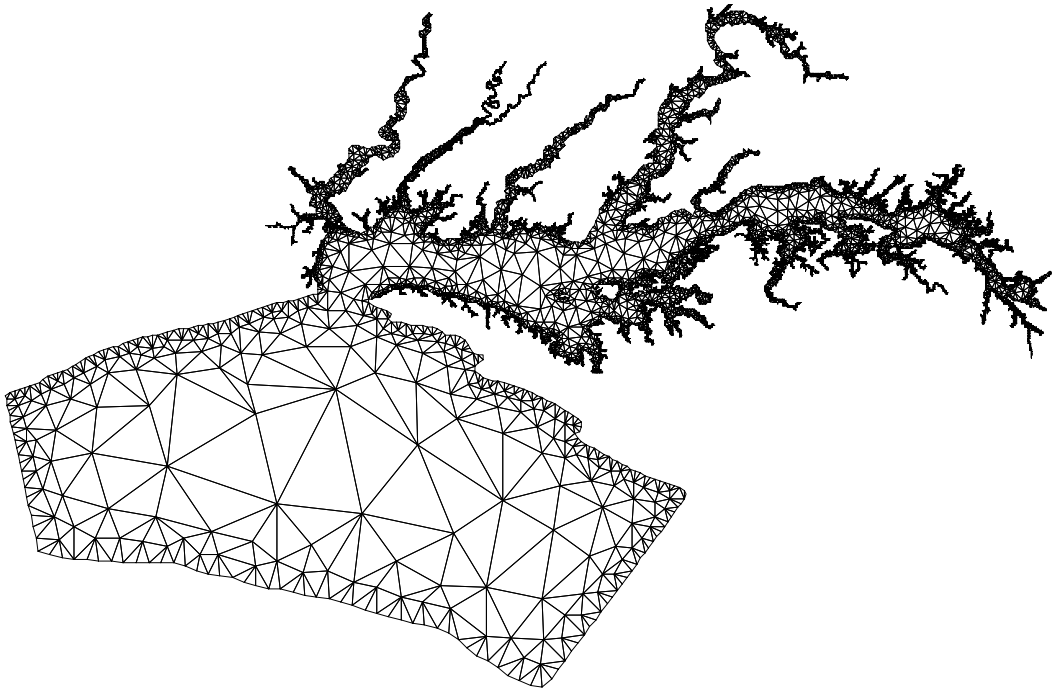

Figure 10: The Chesapeake bay mesh with the minimal angle bound equal to 20°. Steiner points are inserted at the boundary $(c_x + r(1 - \sqrt{2}/\bar{\rho}), c_y)$ of the Type II selection disk: 22558 triangles.

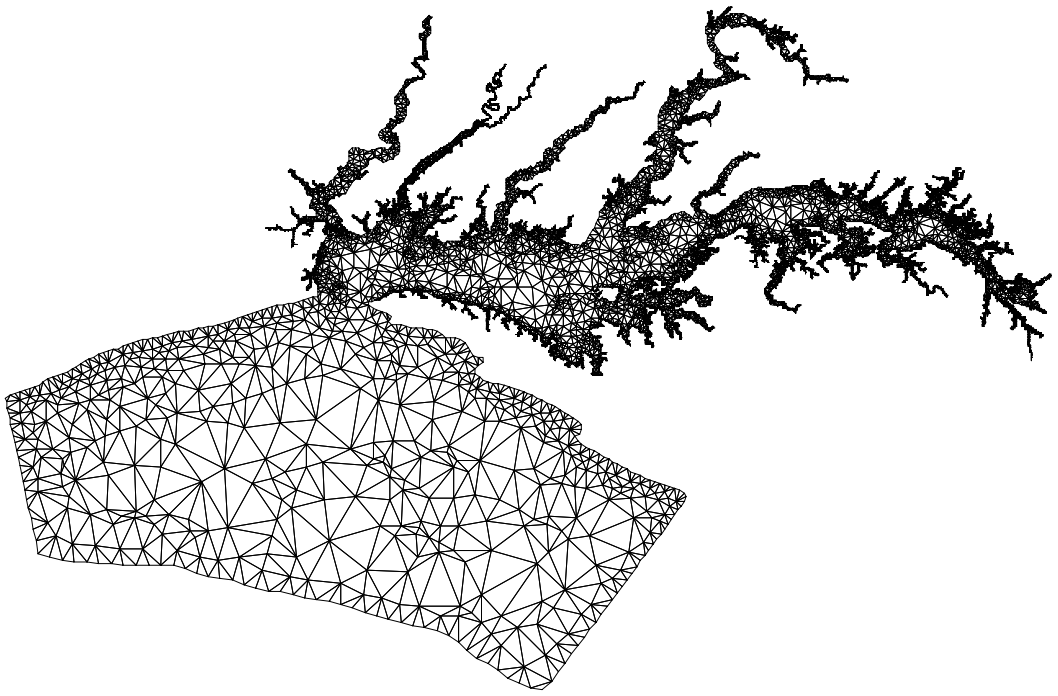

Figure 11: The Chesapeake bay mesh with the minimal angle bound equal to 20°. Steiner points are inserted at the boundary $(c_x + r - \sqrt{2}l, c_y)$ of the Type I selection disks: 25868 triangles.

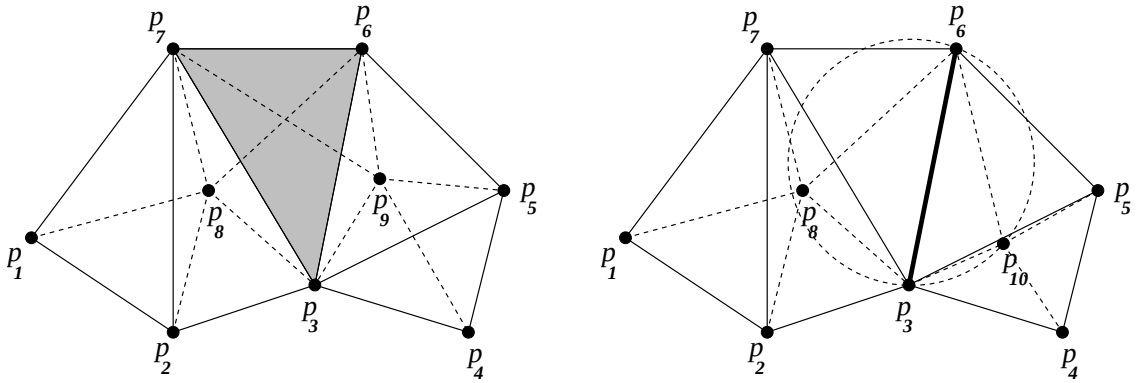

Figure 12: **(Left)** If $\triangle p_3 p_6 p_7 \in \mathcal{C}(p_8) \cap \mathcal{C}(p_9)$, then concurrent insertion of p_8 and p_9 yields a non-conformal mesh. Solid lines represent edges of the initial triangulation, and dashed lines represent edges created by the insertion of p_8 and p_9. Note that the intersection of edges $p_8 p_6$ and $p_9 p_7$ creates a non-conformity. **(Right)** If edge $p_3 p_6$ is shared by $\mathcal{C}(p_8) = \{\triangle p_1 p_2 p_7, \triangle p_2 p_3 p_7, \triangle p_3 p_6 p_7\}$ and $\mathcal{C}(p_{10}) = \{\triangle p_3 p_5 p_6, \triangle p_3 p_4 p_5\}$, the new triangle $\triangle p_3 p_{10} p_6$ can have point p_8 inside its circumdisk, thus, violating the Delaunay property.

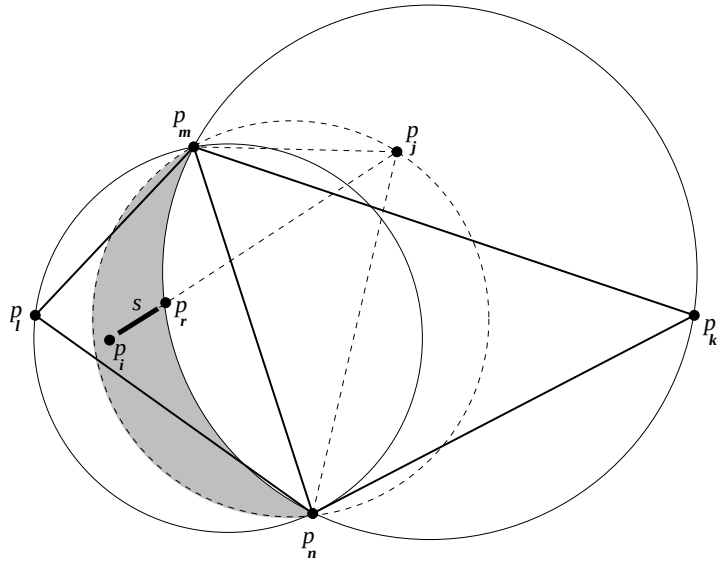

Figure 13: $\triangle p_l p_n p_m \in \mathcal{C}(p_i)$, $\triangle p_k p_m p_n \in \mathcal{C}(p_j)$, $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$, $e(p_m p_n) \in \partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$, $p_i \in \bigcirc(\triangle p_j p_m p_n)$, and $r(\triangle p_k p_m p_n) > r(\triangle p_l p_n p_m)$.

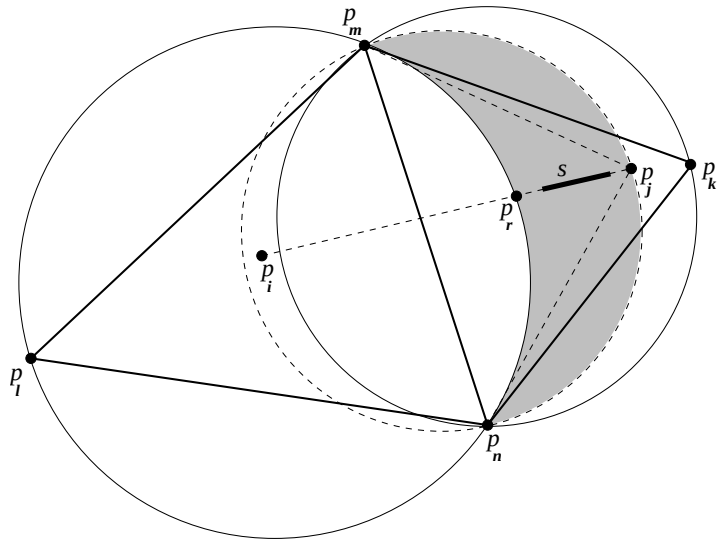

Figure 14: $\triangle p_l p_n p_m \in \mathcal{C}(p_i)$, $\triangle p_k p_m p_n \in \mathcal{C}(p_j)$, $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$, $e(p_m p_n) \in \partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$, $p_i \in \bigcirc(\triangle p_j p_m p_n)$, and $r(\triangle p_k p_m p_n) \leq r(\triangle p_l p_n p_m)$.

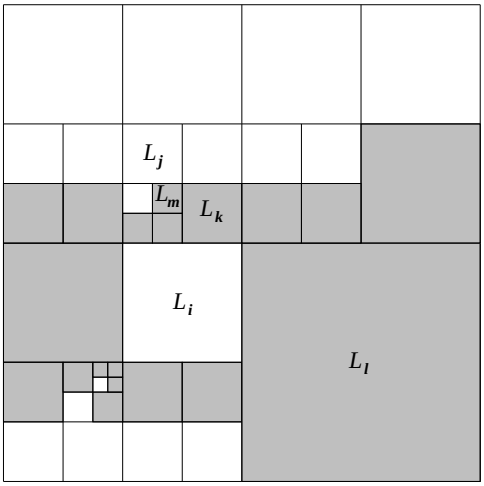

Figure 15: An example of BUF (L_i).

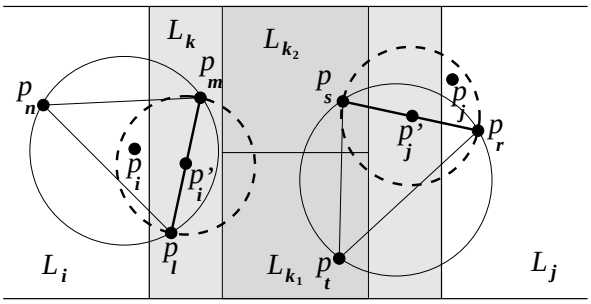

Figure 16: Splitting constrained segments and strong Delaunay-independence.

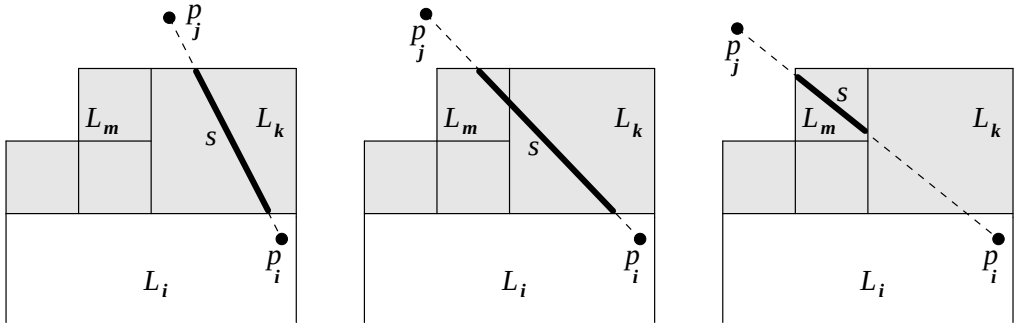

Figure 17: Some possible positions of points p_i and p_j relative to BUF (L_i).

ParallelGeneralizedDelaunayRefinement($\mathcal{X}$, P, g, $\bar{\mathcal{A}}(\cdot)$, $\bar{\rho}$, $f(\cdot)$)
Input: $\mathcal{X}$ is a PSLG which defines Ω
P is the maximum number of compute threads
g is the granularity
$\bar{\mathcal{A}}(\cdot)$ is the triangle area grading function
$\bar{\rho}$ is the upper bound on triangle circumradius-to-shortest edge ratio
$f(\cdot)$ is a deterministic function which returns a specific position within triangle's selection disk
Output: a conforming Delaunay mesh $\mathcal{M}$ respecting $\bar{\mathcal{A}}(\cdot)$ and $\bar{\rho}$

```
1  Let Quadtree be a quadtree, initially consisting of a root node
     which encloses the entire model
2  Construct M, a constrained Delaunay triangulation of X
3  Let RefinementQ = {Leaf ∈ Quadtree | PoorTriangles(Leaf) ≠ ∅}
4  Let p = 0 be the number of spawned threads
5  while RefinementQ ≠ ∅ or p > 0
6    if RefinementQ = ∅ or p = P
7      Wait for a thread to finish refining Leaf
8      p ← p − 1
9      RefinementQ ← RefinementQ ∪
         {L ∈ BUF^2(Leaf) | |PoorTriangles(L)| > 0}
10   else
11     Let Leaf be the leaf on the top of RefinementQ
12     RefinementQ ← RefinementQ \ BUF^2(Leaf)
13     p ← p + 1
14     spawn DelaunayRefinement(X, g, Ā(·), ρ̄, f(·), M, Leaf)
15   endif
16 endwhile
17 return M
```

Figure 18: The Parallel Generalized Delaunay Refinement (PGDR) algorithm executed by the master thread.

GeneralizedDelaunayRefinement($\mathcal{X}$, g, $\bar{\mathcal{A}}(\cdot)$, $\bar{\rho}$, $f(\cdot)$, $\mathcal{M}$, *Leaf*)
Input: See the GPDR algorithm, Figure 18
Output: Locally refined Delaunay mesh $\mathcal{M}$
Locally refined quadtree node *Leaf*

```
1  i_min ← min_{PoorTriangles_i(Leaf)≠∅} i
2  i_max ← i_min + g
3  for j = i_min, ..., i_max
4    while PoorTriangles_j(Leaf) ≠ ∅
5      Let t ∈ PoorTriangles_j(Leaf)
6      p ← f(t)
7      Insert p into M
8      for L ∈ {Leaf} ∪ BUF (Leaf)
9        Update PoorTriangles(L) and Counter(L)
10     endfor
11   endwhile
12 endfor
13 Split Leaf recursively while (29) holds
14 return M, Leaf
```

Figure 19: The algorithm executed by each of the refinement threads.

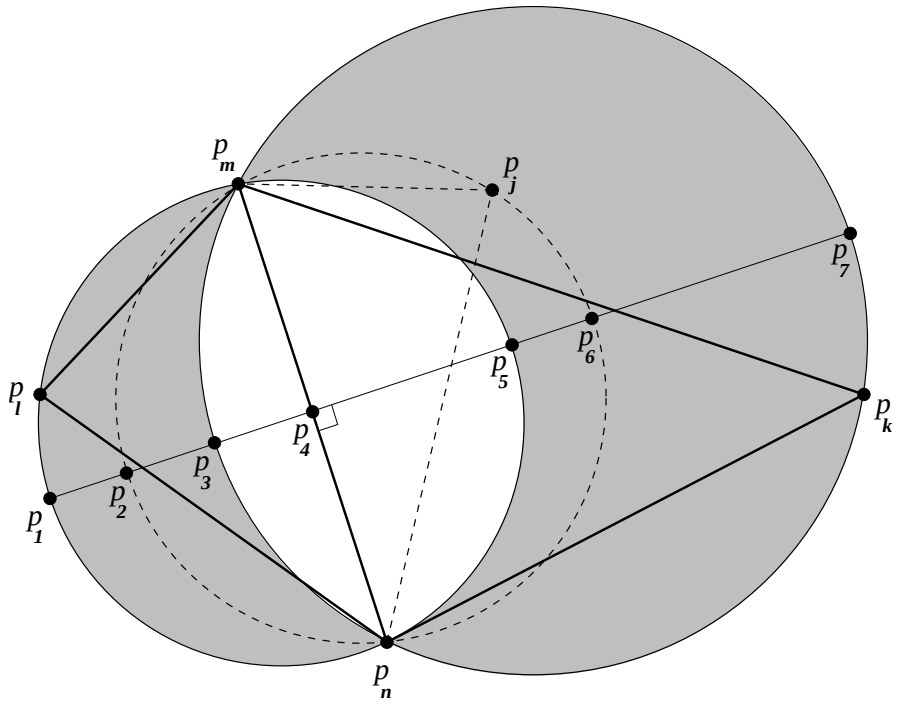

Figure 20: $\triangle p_k p_m p_n \in \mathcal{C}(p_j)$, $\triangle p_l p_n p_m \notin \mathcal{C}(p_j)$.

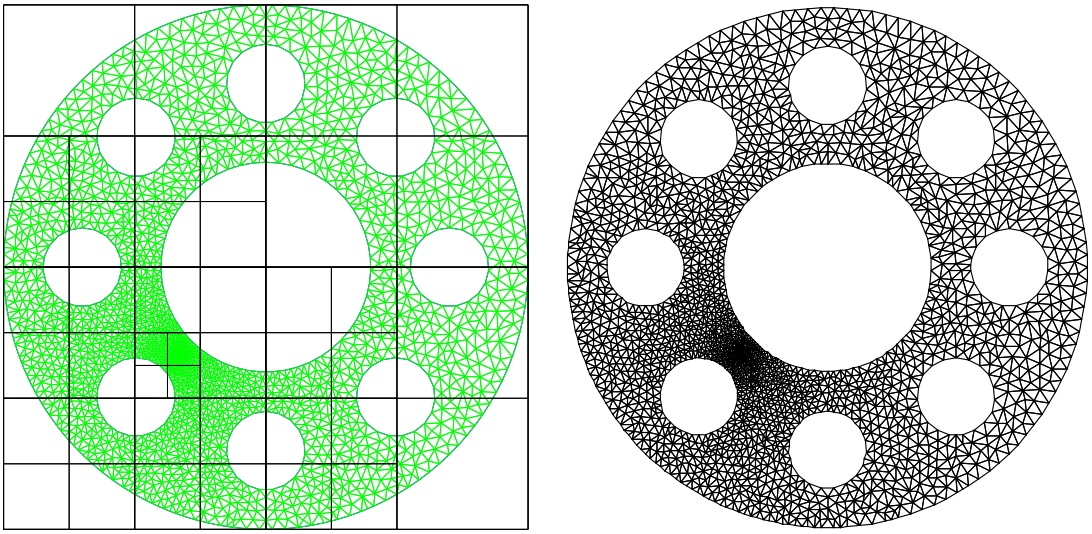

Figure 21: Pipe cross-section model, $\bar{\mathcal{A}}(x,y) = 0.4\sqrt{(x-200)^2+(y-200)^2}+1$. **(Left)** Our parallel refinement algorithm, 4166 triangles. **(Right)** `Triangle` [46], 4126 triangles.

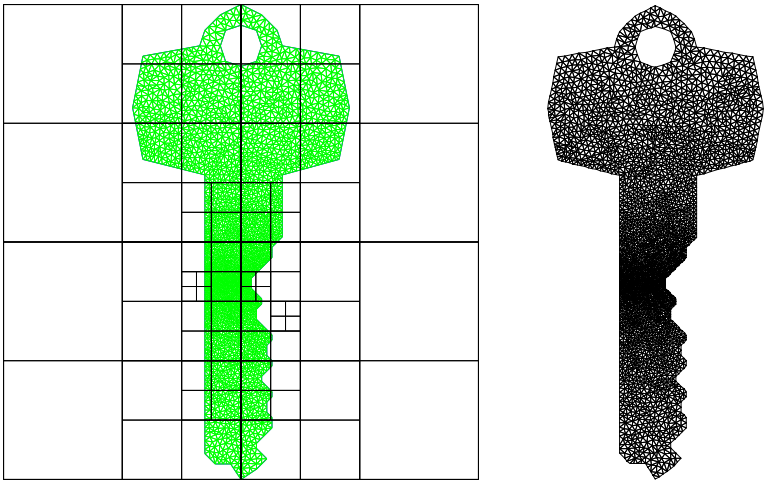

Figure 22: Jonathan Shewchuk's key model, $\bar{\mathcal{A}}(x,y) = 0.02|y-46|+0.1$. **(Left)** Our parallel refinement algorithm, 5411 triangles. **(Right)** `Triangle` [46], 5723 triangles.

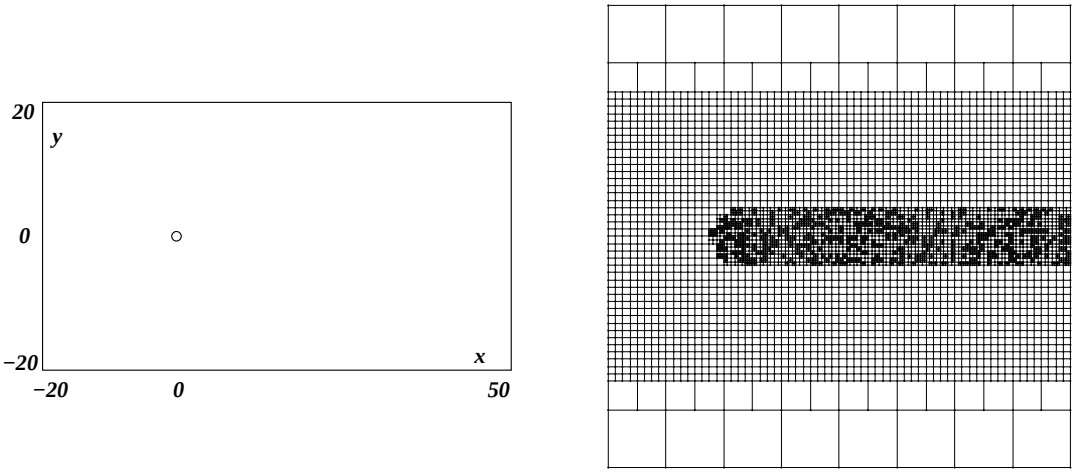

Figure 23: The cylinder flow model. $\bar{\mathcal{A}}(x,y) = 1.2{\cdot}10^{-3}$ if $((x \geq 0) \wedge (|y| < 5)) \vee ((x < 0) \wedge (\sqrt{x^2+y^2}) < 5)$; $\bar{\mathcal{A}}(x,y) = 10^{-2}$, otherwise. Our parallel refinement algorithm produced 1044756 triangles, and `Triangle` [46] produced 1051324 triangles. **(Left)** The input model. **(Right)** The final quadtree. The complete triangulation is not drawn.

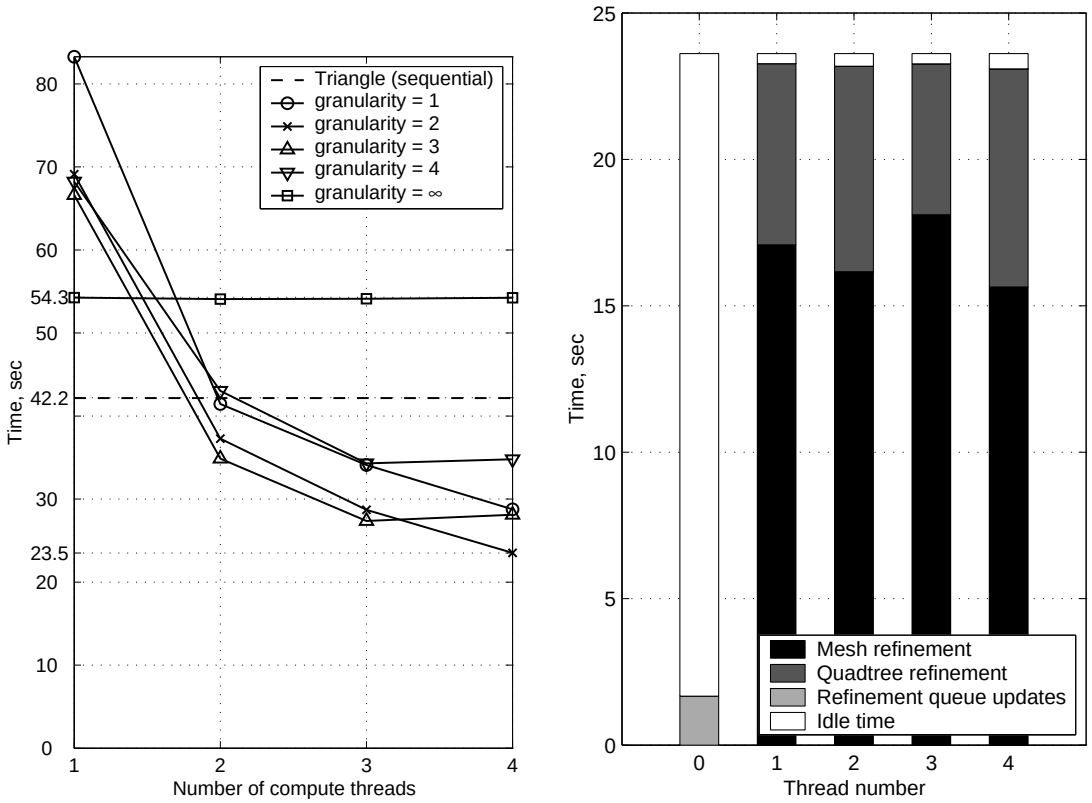

Figure 24: **(Left)** The total running time of the PGDR code, for different granularity values, as the number of compute threads is increased form 1 to 4, compared to `Triangle`.[46] Each point on the graph is the average of 10 measurements. **(Right)** The breakdown of the total PGDR execution time for each of the threads, when the number of compute threads is 4 and granularity is 2. Thread number 0 performs only the management of the refinement queue, and threads 1–4 perform mesh and quadtree refinement. The data correspond to the test with the median total running time.

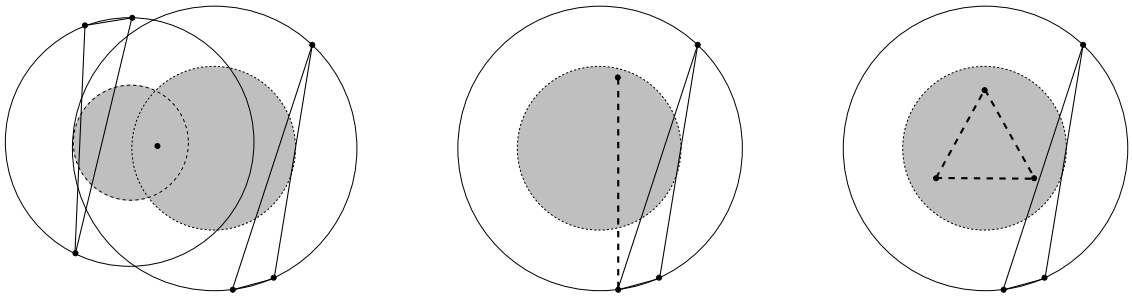

Figure 25: Examples of the approaches for choosing Steiner points within selection disks of skinny triangles.

Table 1: Mesh quality comparison for three point insertion strategies, no area bound is used. θ is the minimal angle bound, n is the number of triangles in the resulting mesh, % is the percentage ratio in the number of triangles over the optimization-based method, $\min A$ and $\max A$ are the minimal and the maximal angles in the entire mesh, A^{ave}_{min} and A^{ave}_{max} are the averages of the minimum and the maximum angle for all triangles.

Point position		$\theta = 10°$		$\theta = 20°$		$\theta = 30°$	
		flow	pipe	flow	pipe	flow	pipe
Circumcenter	n	2173	3033	3153	4651	8758	10655
	%	120.3	106.7	107.5	106.6	138.5	124.1
	$\min A$	10.0	10.0	20.0	20.0	30.0	30.0
	$\max A$	151.5	150.4	128.5	137.7	119.3	119.4
	A^{ave}_{min}	28.1	26.7	35.9	35.9	45.8	45.6
	A^{ave}_{max}	92.5	92.5	88.1	88.0	76.6	76.8
Off-center	n	1906	2941	2942	4411	6175	8585
	%	105.5	103.5	100.3	101.1	97.7	100.0
	$\min A$	10.1	10.0	20.2	20.0	30.0	30.0
	$\max A$	157.0	149.9	133.3	133.7	118.2	119.0
	A^{ave}_{min}	24.4	25.4	34.4	34.7	43.6	43.7
	A^{ave}_{max}	96.2	94.9	87.8	87.1	77.3	77.7
Our example of an optimization-based method	n	1805	2841	2932	4359	6319	8581
	%	100.0	100.0	100.0	100.0	100.0	100.0
	$\min A$	10.0	10.0	20.0	20.0	30.0	30.0
	$\max A$	157.3	152.5	138.3	137.1	119.0	119.5
	A^{ave}_{min}	23.2	24.8	34.3	34.6	44.0	43.6
	A^{ave}_{max}	98.3	96.7	87.9	87.6	77.1	77.8

Table 2: The comparison of the number of triangles generated with the use of circumcenter, off-center, and an optimization-based point insertion strategies, with an area bound. For the cylinder flow model, the area bound is set to $\bar{\mathcal{A}} = 0.01$, and for the pipe cross-section model $\bar{\mathcal{A}} = 1.0$.

Point position	$\theta = 10°$		$\theta = 20°$		$\theta = 30°$	
	flow	pipe	flow	pipe	flow	pipe
Circumcenter	219914	290063	220509	289511	228957	294272
Off-center	219517	290057	220479	289331	226894	295644
Our example of an optimization-based method	219470	289505	220281	289396	226585	294694