Parallel 2D Graded Guaranteed Quality Delaunay Mesh Refinement*

Andrey N. Chernikov[1] and Nikos P. Chrisochoides[1,2,3]

[1]Department of Computer Science
College of William and Mary
McGlothlin-Street Hall
Williamsburg, VA 23185, USA
{ancher,nikos}@cs.wm.edu

[2]Department of Mechanical Engineering, MIT
Boston, MA 02139, USA
[3]Department of Radiology
Harvard Medical School
Boston, MA 02115, USA

Summary. We develop a theoretical framework for constructing guaranteed quality Delaunay meshes in parallel for general two-dimensional geometries. This paper presents a new approach for constructing graded meshes, i.e., meshes with element size controlled by a user-defined criterion. The sequential Delaunay refinement algorithms are based on inserting points at the circumcenters of triangles of poor quality or unacceptable size. We call two points Delaunay-independent if they can be inserted concurrently without destroying the conformity and Delaunay properties of the mesh. The contribution of this paper is three-fold. First, we present a number of local conditions of point Delaunay-independence, which do not rely on any global mesh metrics. 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. Second, we prove that a quadtree, constructed in a specific way, can be used to guide the parallel refinement, so that the points, simultaneously inserted in multiple leaves, are Delaunay-independent. Third, by experimental comparison with the well-known guaranteed quality sequential meshing software, we show that our method does not lead to overrefinement, while matching its quality and allowing for code re-use.

1 Introduction

Parallel 2D mesh generation is still important for some 3D simulations like direct numerical simulations of turbulence in cylinder flows with very large Reynolds numbers [7] and coastal ocean modeling for predicting storm surge and beach erosion in real-time [22]. In both cases, 2D mesh generation is taking place in the xy-plane and it is replicated in the z-direction in the case of cylinder flows or using bathemetric contours in the case of coastal ocean

*This work was supported by NSF grants: EIA-9972853, EIA-0203974, and ACI-0312980

modeling applications. With the increase of the Reynolds number, the size of the mesh grows in the order of $Re^{9/4}$ [13], which motivates the use of parallel mesh generation algorithms. At the same time, the size of the mesh can be somewhat reduced by employing parallel nonuniform mesh refinement, which is the topic of this paper.

Nave, Chrisochoides, and Chew [16] presented a practical provably-good parallel mesh refinement algorithm for polyhedral domains. The approach in [16] allows rollbacks to occur whenever the simultaneously inserted points can potentially lead to an invalid mesh. It is also labor intensive since it requires changing the sequential meshing kernel in order to accommodate for rollbacks and overlapping of computation with communication. In the present paper, we develop a theoretical framework which allows us to guarantee a priori that concurrently inserted points are Delaunay-independent. The elimination of rollbacks leads to two major benefits: savings in the computation time and the possibility to leverage existing sequential Delaunay meshing libraries like the Triangle [18].

Linardakis and Chrisochoides [14] described a Parallel Domain Decoupling Delaunay method for two-dimensional domains, which is capable of leveraging serial meshing codes. However, it can produce only uniform meshes and is based on the Medial Axis Transform, which is very expensive and difficult to compute for three-dimensional geometries. The approach developed in this paper allows to construct nonuniform meshes and 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.

Blelloch, Hardwick, Miller, and Talmor [1] describe a divide-and-conquer projection-based algorithm for constructing Delaunay triangulations of predefined point sets in parallel. The work by Kadow and Walkington [12, 10, 11] extended [2, 1] 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 by a dedicated processor [11].

Edelsbrunner and Guoy [8] define the points x and y as independent if the closures of their *prestars* (or *cavities* [9]) are disjoint. The approach in [8] does not provide a way to avoid computing the cavities and their intersections for all candidate points, which is very expensive. Spielman, Teng, and Üngör [20] presented the first theoretical analysis of the complexity of parallel Delaunay refinement algorithms. In [21] the authors developed a more practical algorithm.

In [5] we presented a theoretical framework and the experimental evaluation of a parallel algorithm for constructing uniform guaranteed quality Delaunay meshes. 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 certain regions will be Delaunay-independent. In

this paper, we build upon the ideas presented in [4] to produce non-uniform (graded) meshes. The non-trivial differences with [5] lie in the introduction of new, local point independence conditions, and in the dynamic construction of a quadtree with leaf size reflecting the local mesh density.

A more extensive review of parallel mesh generation methods can be found in [6].

2 Parallel Refinement Theory

In this section, we develop local Delaunay-independence conditions and show how quadtree leaves can be used to select subsets of circumcenters for concurrent insertion. We extend our previous work [5] 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.

2.1 Terminology and Notation

We will denote point number i as p_i and the triangle with vertices p_i, p_j, and p_k as $\triangle(p_ip_jp_k)$. When the vertices of a triangle are irrelevant, we will write simply $\triangle_r$. An edge of a triangle will be denoted as $e(p_ip_j)$ and a line segment connecting two arbitrary points as $\mathcal{L}(p_ip_j)$. Let us call the open disk corresponding to a triangle's circumcircle its *circumdisk*. We will use symbols $\bigcirc(\triangle(p_ip_jp_k))$, $\odot(\triangle(p_ip_jp_k))$, and $r(\triangle(p_ip_jp_k))$ to represent the circumdisk, circumcenter, and circumradius of $\triangle(p_ip_jp_k)$, respectively.

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 will use a *Planar Straight Line Graph* (PSLG) [18] 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 segments) in the final mesh.

The applications that use Delaunay meshes often impose two constraints on 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 [15, 19]) and an upper bound on the element area. The former 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 $\Delta(x, y) : \mathbb{R}^2 \to \mathbb{R}^1$. As a special case, the grading function can also be constant: $\Delta(x, y) = \bar{\Delta}$.

Typically, a mesh generation procedure starts with constructing an initial mesh, which conforms to the input vertices and segments, and then refines this mesh until the constraints are met. In this paper, we focus on parallelizing the Delaunay refinement stage, which is usually the most memory- and computation-expensive. The general idea of Delaunay refinement is to insert points in the circumcenters of triangles that violate the required bounds, until there are no such triangles left. We will extensively use the notion of *cavity* [9]

which is the set of triangles in the mesh whose circumdisks include a given point p_i. We will denote $\mathcal{C}(p_i)$ to be the cavity of p_i and $\partial\mathcal{C}(p_i)$ to be the set of edges which belong to only one triangle in $\mathcal{C}(p_i)$, i.e., external edges.

For our analysis, we will use the Bowyer-Watson (B-W) point insertion algorithm [3, 23], 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 $\mathcal{M} = (V, T)$ and $\mathcal{M}' = (V', T')$ represent the mesh before and after the insertion of p_i, respectively. The set of newly created triangles forms a *ball* [9] 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 [19, 17]. A vertex p is said to *encroach upon* a segment s, if it lies within the open diametral disk of s [17]. 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 [17], and a cavity of the segment's midpoint is constructed and triangulated as before.

We will use the terms *triangulation* and *mesh* interchangeably, depending on the context.

2.2 Delaunay-independent Points

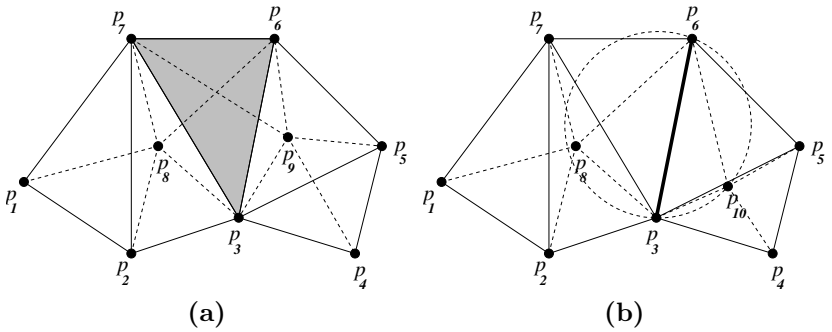


Fig. 1. **(a)** If $\triangle p_3 p_6 p_7 \in \mathcal{C}(p_8) \cap \mathcal{C}(p_9)$, then concurrent insertion of p_8 and p_9 yields a non-conformal mesh. Solid lines represent edges of the initial triangulation, and dashed lines represent edges created by the insertion of p_8 and p_9. Note that the intersection of edges $p_8 p_6$ and $p_9 p_7$ creates a non-conformity. **(b)** 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.

We expect our parallel Delaunay refinement algorithm to insert multiple circumcenters concurrently in such a way that at every iteration the mesh will

be both conformal (i.e., simplicial) and Delaunay. Figure 1 illustrates how the concurrently inserted points can violate one of these conditions.

Definition 1 (Delaunay-independence). *Points p_i and p_j are* Delaunay-independent *with respect to mesh $\mathcal{M} = (V, T)$ if their concurrent insertion yields the conformal Delaunay mesh $\mathcal{M}' = (V \cup \{p_i, p_j\}, T')$. 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 2 (Strong Delaunay-independence). *Points p_i and p_j are* strongly Delaunay-independent *with respect to mesh $\mathcal{M} = (V, T)$ iff any pair of points in $\{p_i, p_i'\} \times \{p_j, p_j'\}$ are Delaunay-independent with respect to $\mathcal{M}$.*

2.3 Local Delaunay-Independence Conditions

Lemma 1 (Delaunay-independence criterion I). *Points p_i and p_j are Delaunay-independent iff*

$$\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset, \tag{2}$$

and

$$\forall e(p_m p_n) \in \partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) \ : \ p_i \notin \bigcirc(\triangle(p_j p_m p_n)). \tag{3}$$

Proof. First, $\mathcal{M}' = (V \cup \{p_i, p_j\}, T')$ is conformal iff (2) holds. Indeed, if (2) holds, then considering (1), the concurrent retriangulation of $\mathcal{C}(p_i)$ and $\mathcal{C}(p_j)$ will not yield overlapping triangles, and the mesh will be conformal. Conversely, if (2) does not hold, the newly created edges will intersect as shown in Fig. 1a, and $\mathcal{M}'$ will not be conformal.

Now, we will show that $\mathcal{M}'$ is Delaunay iff (3) holds. The Delaunay Lemma [9] states that iff the empty circumdisk criterion holds for every pair of adjacent triangles, then the triangulation is globally Delaunay. Disregarding the symmetric cases, there are three types of pairs of adjacent triangles $\triangle_r$ and $\triangle_s$, where $\triangle_r \in \mathcal{B}(p_i)$, that will be affected: (i) $\triangle_s \in \mathcal{B}(p_i)$, (ii) $\triangle_s \in T' \setminus \mathcal{B}(p_i) \setminus \mathcal{B}(p_j)$, and (iii) $\triangle_s \in \mathcal{B}(p_j)$. The sequential Delaunay refinement algorithm guarantees that $\triangle_r$ and $\triangle_s$ will be locally Delaunay in the first two cases. In addition, condition (3) ensures that they will be locally Delaunay in the third case. Therefore, the mesh will be globally Delaunay. Conversely, if (3) 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 [5]). *From Lemma 1 it follows that if (2) holds and $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j) = \emptyset$, then p_i and p_j are Delaunay-independent.*

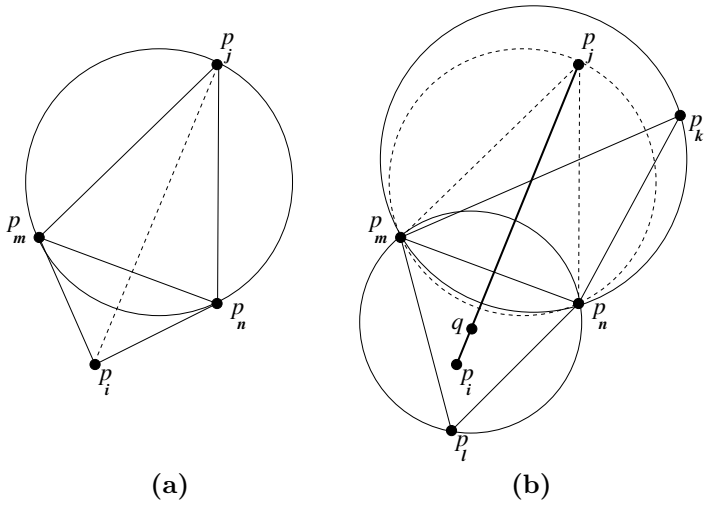


Fig. 2. **(a)** Either $e\,(p_ip_j)$ or $e\,(p_mp_n)$ is locally Delaunay. **(b)** $\bigcirc\,(\triangle\,(p_jp_mp_n))$ cannot include p_i and not include q.

Lemma 2 (Delaunay-independence criterion II). *Points p_i and p_j are Delaunay-independent with respect to mesh $\mathcal{M} = (V, T)$ iff the edge $e\,(p_ip_j)$ does not appear in $\mathcal{M}' = (V \cup \{p_i, p_j\}, T')$.*

Proof. To prove the "if" part, let us recall that an edge e exists in a Delaunay triangulation iff there is an empty open disk whose circle passes through the endpoints of e [19]. This means that, in case $e\,(p_ip_j)$ is not in $\mathcal{M}$, there is no empty open disk whose circle passes through p_i and p_j. This observation has two consequences:

(i) There is no open disk (triangle circumdisk, as a special case), empty of the existing mesh vertices, that includes both p_i and p_j; therefore, condition (2) holds.
(ii) There is no empty open disk, which includes p_i, whose circle passes through p_j. As a special case, there is no such disk whose circle also passes through p_m and p_n; consequently, condition (3) holds.

Thus, p_i and p_j are Delaunay-independent by Lemma 1.

In order to show that the "only if" part of the Lemma holds, we assume that p_i and p_j are Delaunay-independent. Then, by Lemma 1, conditions (2) and (3) hold. Consider Figure 2a. An edge e of a triangulation is either locally Delaunay or is flippable, in which case the edge created by flipping e is locally Delaunay [19]. Since the edge $e\,(p_mp_n)$ is locally Delaunay, the edge $e\,(p_ip_j)$ is not locally Delaunay, and, hence, cannot exist in $\mathcal{M}'$.

Corollary 2 (Sufficient condition of Delaunay-independence II). *Lemma 2 implies that if p_i and p_j are not visible to each other (i.e. the edge $e\,(p_ip_j)$ can-*

not exist in a triangulation of Ω, e.g. it would cross a constrained segment), then p_i and p_j are Delaunay-independent.

Lemma 3 (Sufficient condition of Delaunay-independence III). *Points p_i and p_j are Delaunay-independent if there exists a point $q \in \mathcal{L}(p_i p_j)$ such that*

$$\forall \triangle_s \in T \ : \ q \in \bigcirc(\triangle_s) \implies r(\triangle_s) \leq \frac{1}{2}\|p_i - p_j\|. \tag{4}$$

Proof. First, condition (4) implies that $\mathcal{C}(p_i) \cap \mathcal{C}(p_j) = \emptyset$. Indeed, if there had been a triangle circumdisk that included p_i and p_j, then this circumdisk would have also included q and had radius greater than $\frac{1}{2}\|p_i - p_j\|$, which contradicts (4).

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 $\partial\mathcal{C}(p_i) \cap \partial\mathcal{C}(p_j)$ as depicted on Fig. 2b. $\bigcirc(\triangle(p_j p_m p_n))$ cannot include p_i; otherwise, it would have also included q and had radius greater than $\frac{1}{2}\|p_i - p_j\|$, which contradicts (4). Hence, by Lemma 1, p_i and p_j are Delaunay-independent.

2.4 Quadtree Construction

Definition 3 (Quadtree node). *Let a* quadtree node *be an axis-aligned square $S \subset \mathbb{R}^2$. A quadtree node can be either divided into four smaller nodes of equal size or not divided (in this case it is a* leaf*).*

We will denote the length of the side of square S as $\ell(S)$.

Definition 4 (α-neighborhood). *Let the* α-neighborhood $\mathcal{N}_\alpha(S_i)$ ($\alpha \in \{Left, Right, Top, Bottom\}$) of quadtree leaf S_i *be the set of quadtree leaves that share a side with S_i and are located in the α direction of S_i. For example, in Fig. 3, $S_k \in \mathcal{N}_{Top}(S_i)$ and $S_l \in \mathcal{N}_{Right}(S_i)$.*

Definition 5 (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 6 (Buffer zone). *Let the set of leaves*

$$\mathrm{BUF}(S_i) = \bigcup_\alpha \mathcal{N}_\alpha(S_i) \ \cup \ \{S_m \in \mathcal{N}_{\mathrm{ORT}(\alpha)}(S_k) \ | \ S_k \in \bigcup_\alpha \mathcal{N}_\alpha(S_i)\}$$

be called a buffer zone *of leaf S_i with respect to mesh $\mathcal{M}$ iff*

$$\forall S_n \in \mathrm{BUF}(S_i), \forall \triangle_s \in T \ : \ \bigcirc(\triangle_s) \cap S_n \neq \emptyset \implies r(\triangle_s) < \frac{1}{4}\ell(S_n). \tag{5}$$

Equation (5) 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.

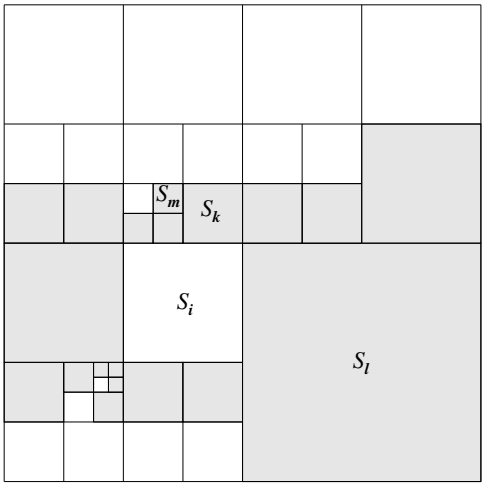


Fig. 3. An example of BUF (S_i).

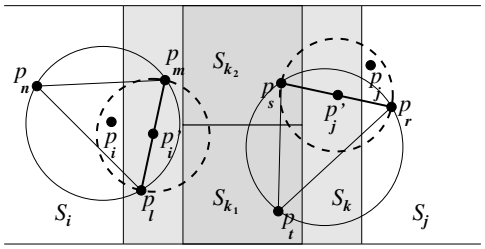


Fig. 4. Splitting constrained segments and strong Delaunay-independence.

Definition 7 (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 arbitrary points $p_i \in R_i$ and $p_j \in R_j$ are strongly Delaunay-independent.*

Lemma 4 (Sufficient condition of square Delaunay-separateness). *If S_i and S_j are quadtree leaves and $S_j \notin$ BUF (S_i), then S_i and S_j are Delaunay-separated.*

Proof. First, for an arbitrary pair of points $p_i \in S_i$ and $p_j \in S_j \notin \mathrm{BUF}(S_i)$, 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, S_i and S_j are Delaunay-separated.

By enumerating all possible configurations of leaves in $\mathrm{BUF}(S_i)$ and grouping similar cases, w.l.o.g. all arrangements can be accounted for using the following argument.

Suppose $\mathcal{L}(p_i p_j)$ intersects the common boundary of S_i and $S_k \in \mathcal{N}_{Top}(S_i) \subset \mathrm{BUF}(S_i)$ (Fig. 5).

(i) If $\mathcal{L}(p_i p_j)$ intersects the upper boundary of S_k (Fig. 5a), then, from (5) and the fact that $\ell(S_k) < \|p_i - p_j\|$, any point $q \in \mathcal{L}(p_i p_j) \cap S_k$ will satisfy (4). Therefore, by Lemma 3, p_i and p_j are Delaunay-independent.

(ii) Otherwise, let $\mathcal{L}(p_i p_j)$ intersect the left boundary of S_k and $S_m \in \mathcal{N}_{Left}(S_k) \subset \mathrm{BUF}(S_i)$ be the leaf adjacent to this boundary at the point of intersection. $\mathcal{L}(p_i p_j)$ can intersect either the upper boundary of S_m (Fig. 5b) or the left boundary of S_m (Fig. 5c). In both cases, $\ell(S_m) < \|p_i - p_j\|$, any point $q \in \mathcal{L}(p_i p_j) \cap S_m$ will satisfy (4), and p_i and p_j are Delaunay-independent by Lemma 3.

Now, suppose p_i and p_j encroach upon constrained edges $e(p_l p_m)$ and $e(p_r p_s)$, respectively (Fig. 4). 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. $p_i', p_j' \in S_k \in \mathrm{BUF}(S_i)$. Since the diametral disk of an edge has the smallest radius among all disks whose circle passes through the endpoints of an edge, then $r(e(p_l p_m)) \le r(\triangle(p_l p_m p_n)) < \frac{1}{4}\ell(S_k)$ and $r(e(p_r p_s)) \le r(\triangle(p_r p_s p_t)) < \frac{1}{4}\ell(S_k)$. Therefore, $\|p_i' - p_j'\| > \frac{1}{2}\ell(S_k)$. By constructing imaginary buffer squares S_{k_1} and S_{k_2} as shown on

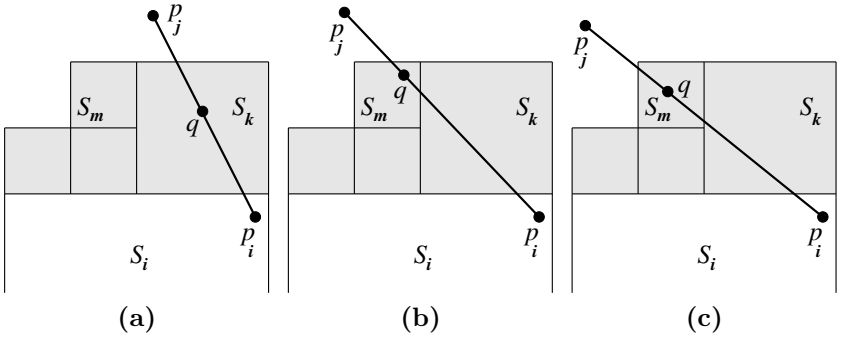
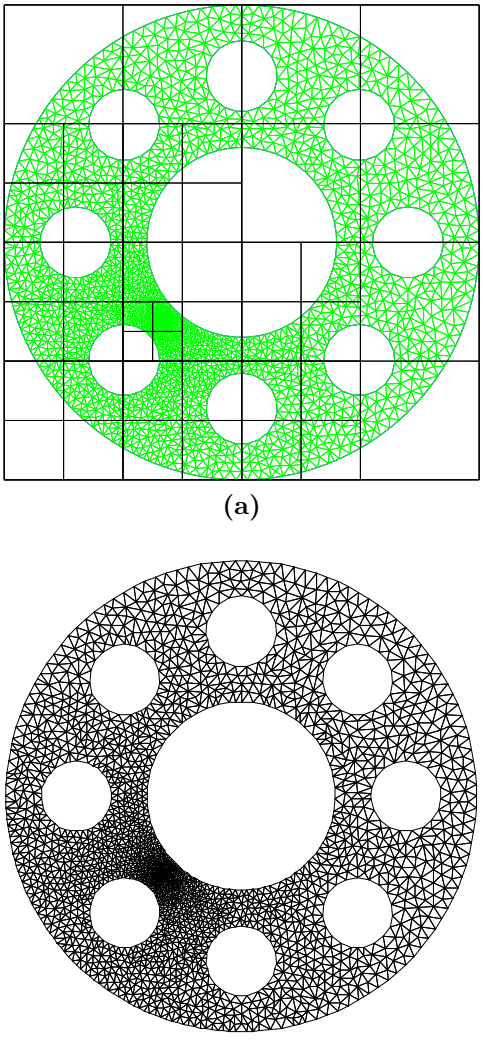


Fig. 5. Some possible positions of points p_i and p_j relative to $\mathrm{BUF}(S_i)$.

Fig. 4, we can still satisfy condition (4), which guarantees that p'_i and p'_j are Delaunay-independent by Lemma 3.

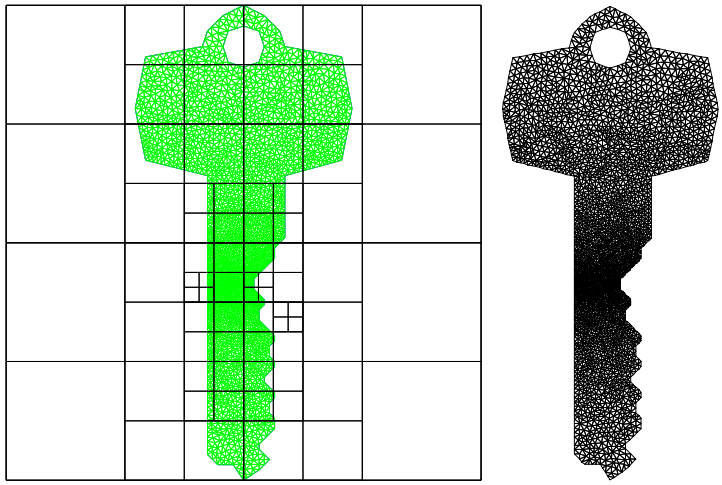
3 Experiments



(a)

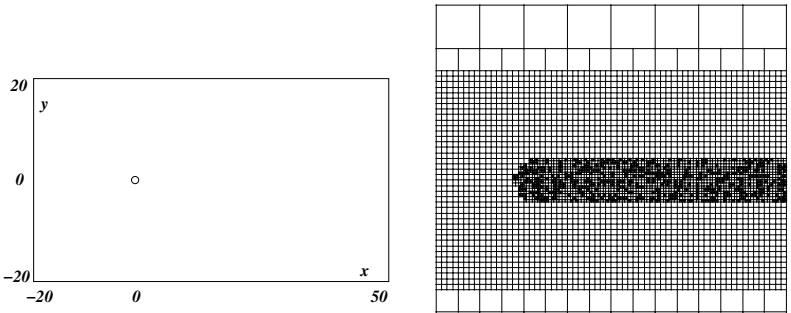
(b)

Fig. 6. Pipe cross-section model, $\Delta(x, y) = 0.4\sqrt{(x-200)^2 + (y-200)^2} + 1$. **(a)** Our parallel refinement algorithm, 4166 triangles. **(b)** The Triangle [18], 4126 triangles.



(a) **(b)**

Fig. 7. Jonathan Shewchuk's key model, $\Delta(x, y) = 0.02|y - 46| + 0.1$. **(a)** Our parallel refinement algorithm, 5411 triangles. **(b)** The Triangle [18], 5723 triangles.



(a) **(b)**

Fig. 8. The cylinder flow model. $\Delta(x, y) = 1.2 \cdot 10^{-3}$ if $((x \geq 0) \wedge (y < 5)) \vee ((x < 0) \wedge (\sqrt{x^2 + y^2}) < 5)$; $\Delta(x, y) = 10^{-2}$, otherwise. Our parallel refinement algorithm produced 1044756 triangles, and the Triangle [18] produced 1051324 triangles. **(a)** The input model. **(b)** The final quadtree. The complete triangulation is not drawn.

Figures 6 and 7 compare the meshes produced by our implementation and the Triangle library [18] for a pipe cross-section and a key. Figure 8 also shows the initial geometry and the quadtree produced by our algorithm for the cylinder flow problem, which is similar to the model used in [7]. 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 are used as follows. If (x_i, y_i) is the centroid of the triangle $\triangle_i$, then the area of $\triangle_i$ has to be less than $\Delta(x_i, y_i)$. 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 the Triangle [18].

4 Conclusions

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 overrefinement.

We are currently working on the extension of the proposed approach to three dimensions. While the quadtree immediately generalizes to the octree, the properties of 3D cavities require further study.

5 Acknowledgments

We thank the anonymous reviewers for helpful comments.

References

1. G. E. Blelloch, J. Hardwick, G. L. Miller, and D. Talmor. Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica*, 24:243–269, 1999.
2. G. E. Blelloch, G. L. Miller, and D. Talmor. Developing a practical projection-based parallel Delaunay algorithm. In *12th Annual Symposium on Computational Geometry*, pages 186–195, 1996.
3. A. Bowyer. Computing Dirichlet tesselations. *Computer Journal*, 24:162–166, 1981.
4. A. N. Chernikov and N. P. Chrisochoides. Parallel guaranteed quality planar Delaunay mesh generation by concurrent point insertion. In *14th Annual Fall Workshop on Computational Geometry*, pages 55–56. MIT, Nov. 2004.

5. A. N. Chernikov and N. 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. ACM Press, 2004.
6. N. P. Chrisochoides. A survey of parallel mesh generation methods. Technical Report BrownSC-2005-09, Brown University, 2005. To appear in Numerical Solution of Partial Differential Equations on Parallel Computers (eds. Are Magnus Bruaset, Petter Bjorstad, Aslak Tveito).
7. S. Dong, D. Lucor, and G. E. Karniadakis. Flow past a stationary and moving cylinder: DNS at Re=10,000. In *2004 Users Group Conference (DOD_UGC'04)*, pages 88–95, 2004.
8. H. Edelsbrunner and D. Guoy. Sink-insertion for mesh improvement. In *Proceedings of the Seventeenth Annual Symposium on Computational Geometry*, pages 115–123. ACM Press, 2001.
9. P.-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing. Application to Finite Elements*. HERMES, 1998.
10. C. Kadow. Adaptive dynamic projection-based partitioning for parallel Delaunay mesh generation algorithms. In *SIAM Workshop on Combinatorial Scientific Computing*, Feb. 2004.
11. C. Kadow. *Parallel Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, 2004.
12. C. Kadow and N. Walkington. Design of a projection-based parallel Delaunay mesh generation and refinement algorithm. In *Fourth Symposium on Trends in Unstructured Mesh Generation*, July 2003. http://www.andrew.cmu.edu/user/sowen/usnccm03/agenda.html.
13. G. Karnriadakis and S. Orszag. Nodes, modes, and flow codes. *Physics Today*, 46:34–42, 1993.
14. L. Linardakis and N. Chrisochoides. Parallel domain decoupling Delaunay method. *SIAM Journal on Scientific Computing*, in print, accepted Nov. 2004.
15. G. L. Miller, D. Talmor, S.-H. Teng, and N. Walkington. A Delaunay based numerical method for three dimensions: Generation, formulation, and partition. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, pages 683–692. ACM Press, May 1995.
16. D. Nave, N. Chrisochoides, and L. P. Chew. Guaranteed–quality parallel Delaunay refinement for restricted polyhedral domains. *Computational Geometry: Theory and Applications*, 28:191–215, 2004.
17. J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548–585, 1995.
18. J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Proceedings of the First workshop on Applied Computational Geometry*, pages 123–133, Philadelphia, PA, 1996.
19. J. R. Shewchuk. *Delaunay Refinement Mesh Generation*. PhD thesis, Carnegie Mellon University, 1997.
20. D. A. Spielman, S.-H. Teng, and A. Üngör. Parallel Delaunay refinement: Algorithms and analyses. In *Proceedings of the Eleventh International Meshing Roundtable*, pages 205–217, 2001.
21. D. A. Spielman, S.-H. Teng, and A. Üngör. Time complexity of practical parallel Steiner point insertion algorithms. In *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 267–268. ACM Press, 2004.

22. R. A. Walters. Coastal ocean models: Two useful finite element methods. *Recent Developments in Physical Oceanographic Modelling: Part II*, 25:775–793, 2005.
23. D. F. Watson. Computing the n-dimensional Delaunay tesselation with application to Voronoi polytopes. *Computer Journal*, 24:167–172, 1981.