Simultaneous Refinement and Coarsening for Adaptive Meshing *

Xiang-Yang Li Shang-Hua Teng Alper Üngör

Abstract. *In numerical simulation of the combustion process and microstructural evolution, we need to consider the adaptive meshing problem for a domain that has a moving boundary. During the simulation, the region ahead of the moving boundary needs to be refined (to satisfy stronger numerical conditions), and the submesh in the region behind the moving boundary should be coarsened (to reduce the mesh size). We present a unified scheme for simultaneously refining and coarsening a mesh. Our method uses sphere packings and guarantees that the resulting mesh is well-shaped and is within a constant factor of the optimal possible in the number of mesh elements. We also present several practical variations of our provably good algorithm.*

keywords. adaptive meshing, coarsening, refinement, moving boundary, sphere packing, Delaunay triangulation.

1 Introduction

In numerical simulation of many problems, we need to handle an evolving mesh which changes its size and topology as a function of time or the number of iterations of a numerical procedure. There are two basic scenarios:

- **Adaptive refinement (based on** *a posterior* **error analysis)**: In numerical simulation of time-independent problems, an iterative procedure is often used. It first generates a mesh based on *a priori* estimates of the local mesh density, solves the numerical equations defined on the initial mesh, and then, based on the *a posterior* error analysis, adaptively refines the mesh and solves the new numerical equations. It could repeats these steps until an accurate solution is obtained.
- **Dynamic meshing with a moving boundary**: In numerical simulation of time-dependent problems such as the combustion process and microstructural evolution, we need to consider the adaptive meshing problem for a domain that has a moving boundary, in which, the mesh should be dynamically changed to be effective for the next time-step simulation.

In both cases, submeshes in some parts of the domain need to be refined, while in some other parts need to be coarsened. For example, the moving boundary of a time-dependent problem could

*Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801. Supported in part by an NSF CAREER award (CCR-9502540), an Alfred P. Sloan Research Fellowship, and the U.S. Department of Energy through the University of California under Subcontract number B341494 (DOE ASCI).

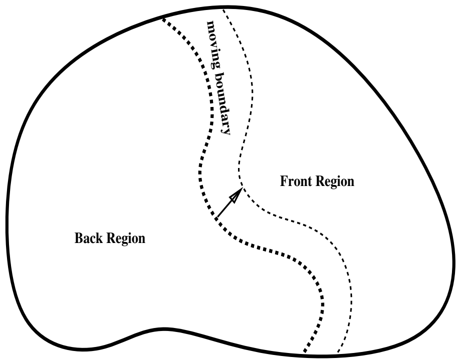


Figure 1: Domain with a moving boundary.

divide the domain into two regions: the front region and the back region. See Fig. 1. During the simulation, numerical conditions in the front region become stronger, requiring its submesh to be refined. In contrast, the submesh in the back region should be coarsened in order to reduce the mesh size. Therefore, it is desirable to have an efficient algorithm which can simultaneously refine and coarsen a mesh.

We present a unified framework for this problem. It is built upon previous works in mesh generation and coarsening [2, 10, 7, 11]. It has a mesh density estimation procedure which computes the desirable mesh spacing using the structure of the current mesh. It then uses a sphere packing based meshing algorithm to generate a mesh that satisfies the estimated mesh density. At a high level, given a mesh M and an updated numerical condition, it applies the following steps to construct a new mesh M'.

1. Estimate the density of M' based on M and the updated numerical condition.
2. Based on the information of the mesh density, compute the new spacing at each mesh point in M;
3. Determine the coarsening factor of each mesh point, referred as a C-point, whose new spacing is larger than the previous one (such as for mesh points in the back region), and the refining factor of each mesh point, referred as an R-point, whose new spacing is smaller than the previous one (such as for mesh points in the front region);
4. Properly scale up the spheres of all C-points and scale down the spheres of all R-points, and fill in the gaps among the shrunk spheres with new spheres of proper sizes;
5. From the sphere system, construct the point set of the new mesh;
6. Use Delaunay triangulation to generate the new mesh M'.

We will show that our method guarantees that the resulting mesh is well-shaped and is of a size that is within a constant factor of the optimal possible. In addition, our algorithm minimizes the number of new spheres needed for the sphere packing, and hence retains the structure of the original mesh as much as possible. We also present some practical variations of our algorithm.

Section 2 introduces the Evolving Mesh Problem, which provides a general framework for studying mesh refinement and coarsening. Some useful notation and basic definitions are also reviewed. Section 3 describes our method for simultaneous coarsening and refinement. We show that the time complexity of our algorithm for the Evolving Mesh Problem is $O(n \log n)$. Section 4 gives a proof of the size and quality bound of our algorithm. Several more practical versions of our algorithms are presented in Section 5. In Sections 6 and 7, we conclude the paper with some experimental results.

2 The Evolving Mesh Problem

The *Evolving Mesh Problem* (EMP) is more general than the dynamic meshing problem that has a moving boundary. To introduce EMP, we first review some basic definitions of unstructured meshes.

2.1 Well-Shaped Meshes

A *mesh* M is a discretization of a domain Ω into a collection of simple elements. We consider unstructured meshes which have varying local topology and spacing, and in which each element is a simplex, i.e., a triangle in 2D or a tetrahedron in 3D. The use of unstructured meshes is necessary for simulating irregular engineering problems with fewer mesh elements [2, 8, 10].

Numerical approximation errors depend on the *quality* of the mesh, while the time and the space required by numerical algorithms are a function of the number of mesh elements. To properly approximate a continuous function, in addition to the conditions that a mesh must conform to the boundaries of the region and be fine enough, each individual element of the mesh must be *well-shaped*. A common shape criterion for the mesh elements is the condition that the angles of each element are not too small, or the aspect ratio of each element is bounded [1, 2, 13]. In this paper, we measure the quality of a triangular element by the *radius-edge ratio* as defined in [8, 9].

Definition 2.1 (radius-edge ratio) *The radius-edge ratio of a simplex is the ratio of the circumradius to the length of the shortest edge of the simplex.*

In two dimensions, the radius-edge ratio is closely related with the smallest angle; the radius-edge ratio is bounded from above by a constant if and only if the smallest angle is bounded below by a constant. Based on this quality measure, we can define well-shaped meshes as the following.

Definition 2.2 (α-well-shaped mesh) *A mesh M is* α-well-shaped *for a constant $\alpha > 1$ if the minimum radius-edge ratio over all of its elements is bounded from above by α.*

2.2 Spacing Functions

A spacing function specifies how fine a mesh should be at a particular region. This function can be derived from the previous numerical results and or from the local geometry feature. In the case of a surface mesh, the spacing function could be determined by local curvatures. Given a well-shaped mesh M over a domain Ω, there are several ways to describe its spacing function.

Definition 2.3 (Edge-length function, el_M) *For each point $x \in \Omega$, $el_M(x)$ is equal to the length of the longest edge of all mesh elements that contain x. Note that points on the lower dimensional faces of a simplex are contained in more than one element.*

Definition 2.4 (Nearest-neighbor function, nn_M) *Let x be a point in Ω. Then $nn_M(x)$ is equal to the distance to the second closest mesh point including x in the case that x is also a mesh point.*

Lemma 2.1 ([8]) *If M is α-well-shaped, then there exist constants c_1 and c_2 depending only on α such that for each point $x \in \Omega$,*

$$c_1 el_M(x) \leq nn_M(x) \leq c_2 el_M(x). \quad (1)$$

As shown in [8, 11], the spacing function for a well-shaped mesh should be smooth in the sense that it changes slowly as a function of distance. The following property specifies the smoothness of a function.

Definition 2.5 (α-Lipschitz function) *A function f is* Lipschitz with a constant α *if for any two points x, y in the domain, $|f(x) - f(y)| \leq \alpha||x - y||$.*

2.3 The Evolving Mesh Problem

Definition 2.6 (The Evolving Mesh Problem (EMP)) *The input to the problem has two parts: (1) a well-shaped mesh M and (2) a list of positive reals δ, one for each mesh point, i.e., associated with each mesh point p is a real number $\delta(p)$, such that $l \leq \delta(p) \leq L$ for constants $0 < l < 1$ and $L > 1$.* [1]

We would like to construct a new mesh M' with the following properties:

- *For each mesh point p in M, $nn_{M'}(p) \leq \delta(p) nn_M(p)$;*
- *M' is well-shaped; and*
- *the size of M' is as small as possible.*

For each mesh point $p \in M$, if $\delta(p) > 1$, then it is a C-point (where C stands for coarsening); if $\delta(p) < 1$, then it is a R-point (where R stands for refinement). Our definition of the Evolving Mesh Problem allows some parts of the mesh to be coarsened while some others to be refined.

To model the dynamic meshing problem with a moving boundary by EMP, we can define δ as a function of the moving boundary. For example, we can define the new spacing of a mesh point by applying a proper monotonic function to the distance from it to the closest point on the moving boundary. EMP is more general in the sense that it does not require any correlation in the change of δ among mesh points.

EMP is closely related with adaptive mesh generation. In the literature, adaptive mesh generation is rather a general term. It has been used, in the context of mesh generation, as to adapt the mesh to the domain geometry or to the *a priori* error estimates. Most often, it has been used to refer to the problem for adaptively refining a mesh based on a newly derived error bound. Here we would like to use EMP to emphasize the problem of simultaneously refining and coarsening a mesh.

[1] The constant l defines the maximum degree of the refinement. The smaller the value l, the more the mesh can be refined. In practice, l is a reasonably large constant, such as $1/10$.

3 An Adaptive Scheme for Evolving Meshes

In this section, we present our basic algorithm for EMP. Our approach extends the spacing-function-based coarsening technique of Miller, Talmor, and Teng [7] to simultaneously refine and coarsen a mesh. Notice, however, the algorithm of [7] does not directly apply to EMP. See the end of Section 4 for a detailed discussion.

In our algorithm, we would like to use the structure of the current mesh M as much as possible and as efficiently as possible.

3.1 A New Spacing Function

For each mesh point p in M, we define a local spacing function $f_p(x)$ as

$$f_p(x) = \delta(p) nn_M(p) + ||x - p||. \tag{2}$$

This spacing function increases with the distance, and has a Lipschitz constant 1. The global spacing $f(x)$ is then given as

$$f(x) = \min_{p \in M} f_p(x). \tag{3}$$

In other words, f is the lower envelope of all local spacing functions. It is easy to show that f is 1-Lipschitz [7].

The following lemma states the relationship between the new spacing function and the nn_M derived from the previous mesh M.

Lemma 3.1 *For any mesh point p, if $\delta(p) \leq 1$, then $f(p) = \delta(p) nn_M(p)$; if $\delta(p) \geq 1$, then $nn_M(p) \leq f(p) \leq \delta(p) nn_M(p)$.*

Proof: By definition, $f(p) = min(min_{q \neq p} f_q(p), f_p(p))$, and $f_p(p) = \delta(p) nn_M(p)$. We first prove that $min_{q \neq p} f_q(p) \geq nn_M(p)$. For all $q \neq p$,

$$\begin{aligned} f_q(p) &= \delta(q) nn_M(q) + ||q - p|| \\ &\geq ||q - p|| \\ &\geq nn_M(p). \end{aligned}$$

The last inequality follows from the definition of nn_M. Hence, $min_{q \neq p} f_q(p) \geq nn_M(p)$.

If $\delta(p) \leq 1$, then $f(p) = min(min_{q \neq p} f_q(p), f_p(p)) = \delta(p) nn_M(p)$. Otherwise, $f(p) \leq f_p(p) = \delta(p) nn_M(p)$, and $f_p(p) = \delta(p) nn_M(p) \geq nn_M(p)$ implying $f(p) \geq nn_M(p)$. □

Hence, the new spacing function satisfies that $f(p) \geq l \cdot nn_M(p)$, i.e., $nn_M(p) \leq f(p)/l$, where l is the lower bound of the δ value for all mesh points. Note that $f(p) \leq \delta(p) nn_M(p)$ implies $f(p) \leq L * nn_M(p)$ for every mesh point p. In addition to the fact that the new spacing function f is 1-Lipschitz, it also has the local similarity property given in the following lemma.

Lemma 3.2 (Local Similarity in a Mesh Element) *For each edge $(p, q) \in M$, there exists a constant c_3 which depends only on the radius-edge ratio α of M and the lower bound l of δ, such that*

$$f(p) \leq c_3 f(q). \tag{4}$$

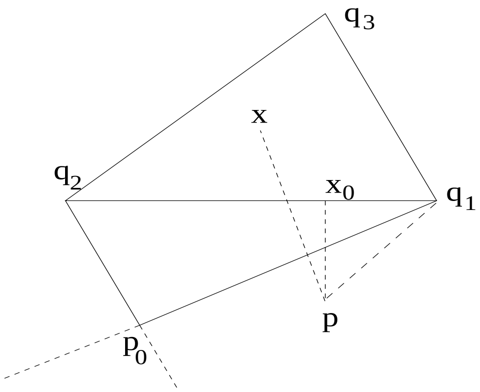


Figure 2: The distance ratio $||p-x||/min_{1\le i\le 3}||p-q_i||$ is at least $\tan(\theta)$, where θ is the lower bound on the angle of mesh elements.

Proof: The fact that f is 1-Lipschitz function implies:

$$\begin{aligned} f(p) &\le f(q) + ||p-q|| \\ &\le f(q) + nn_M(q)/c_1 \\ &\le f(q) + f(q)/(c_1 l) \\ &= (1+1/(c_1 l)) f(q). \end{aligned}$$

The second inequality follows from Lemma 2.1: $||p-q|| \le el(q) \le nn_M(q)/c_1$. The third inequality is from Lemma 3.1. Hence $f(p) \le c_3 f(q)$, where $c_3 = 1+1/(c_1 l)$. Note that the constant c_1 depends only on α. □

Similarly, for any point x in a triangle $\triangle q_1q_2q_3$, we have $f(x) \le c_3 f(q_i)$, for $i=1,2,3$. Note that the above lemma also implies that $f(p) \ge 1/c_3 f(q)$, for any edge (p,q) of the mesh.

Lemma 3.3 (Bounded Distance Ratio) *Let $\triangle q_1q_2q_3$ be a triangle of M. Let p be a mesh point other than q_1,q_2,q_3. Let x be a point inside the triangle. Then there exists a constant c_4, depending only on the smallest angle θ of M, such that*

$$||p-x||/min_{1\le i\le 3}||p-q_i|| \ge c_4. \qquad (5)$$

Proof: First of all, the nearest point in the triangle to p must be on the boundary of the triangle. There are two cases:

- The nearest point is one of $\{q_1,q_2,q_3\}$; We have $||p-x|| \ge min_{1\le i\le 3}||p-q_i||$, i.e.,

$$||p-x||/min_{1\le i\le 3}||p-q_i|| \ge 1.$$

- The nearest point is a boundary point other than q_1, q_2 or q_3. Without loss of generality, assume q_1q_2 separates p from q_3. Let x_0 be the closest point on the segment q_1q_2 to p. See Figure 2. If p is directly connected to q_1 and q_2 in the mesh, then $||p-x_0||/||p-q_i|| \ge \tan(\theta)$, where $i=1,2$, and θ is the lower bound on the element angle. Otherwise, assume that p_0 is the mesh point other than q_3 that is directly connected to q_1 and q_2 in the mesh. Either

p_0q_1 separates p from q_2 or p_0q_2 separates p from q_1 or both. Without loss of generality, assume that p_0q_2 separates p from q_1. We have $||p-x_0||/||p-q_2|| \geq \tan(\theta)$, which implies that $||p-x_0||/min_{1\leq i\leq 3}||p-q_i||$ is at least $\tan(\theta)$. The lemma then follows from the fact that $||p-x|| \geq ||p-x_0||$.

In both cases, we have

$$||p-x||/min_{1\leq i\leq 3}||p-q_i|| \geq min(1, \tan(\theta)).$$

□

Usually, θ is less than $\pi/4$, which implies that $c_4 = \tan(\theta)$.

Lemma 3.4 (Local Similarity in the Domain) *Let x be a point in a triangle $\triangle q_1q_2q_3$ of M. Then there exists a constant c_5, depending only on the smallest angle θ of M, and the lower bound l on δ, such that*

$$c_5 \leq f(x)/min_{1\leq i\leq 3}f(q_i) \leq c_3. \tag{6}$$

Proof: From the definition of f, there exists a mesh point p such that $f(x) = \delta(p)nn_M(p)+||x-p||$. If p is one of $\{q_1, q_2, q_3\}$, say p is q_1, then

$$\begin{aligned} f(x) &= \delta(q_1)nn_M(q_1) + ||x-q_1|| \\ &\geq \delta(q_1)nn_M(q_1) \\ &\geq f(q_1) \\ &\geq min_{1\leq i\leq 3}f(q_i). \end{aligned}$$

Otherwise, let q be the q_i with the minimum distance to p. We have $f(q) \leq \delta(p)nn_M(p) + ||q-p||$.

If $||x-p|| \geq ||q-p||$, then $f(x) = \delta(p)nn_M(p) + ||x-p|| \geq f(q)$. Otherwise,

$$\begin{aligned} f(x)/f(q) &\geq (\delta(p)nn_M(p) + ||x-p||)/(\delta(p)nn_M(p) + ||q-p||) \\ &\geq ||x-p||/||q-p|| \\ &\geq c_4 \end{aligned}$$

The last inequality is given in Lemma 3.3. In both cases, we have $f(x)/f(q) \geq min(1, c_4)$. From Lemma 3.2, we have $f(q)/min_{1\leq i\leq 3}f(q_i) \geq 1/c_3$. Hence,

$$f(x)/min_{1\leq i\leq 3}f(q_i) \geq min(1, c_4)/c_3.$$

In other words, $c_5 = c_4/c_3$.

The proof of $f(x)/min_{1\leq i\leq 3}f(q_i) \leq c_3$ is same as the proof of Lemma 3.1. □

3.2 Functional-Refining-Functional-Coarsening Algorithm

The basic idea of `Functional-Refining-Functional-Coarsening` *FRFC* is to first compute a maximum spacing function that satisfies the new spacing requirement of the Evolving Mesh Problem. We then make use of the point set of M to construct a sphere packing with respect to the spacing function. Then the new mesh M' is obtained by using Delaunay triangulation.

Let $B(x, r)$ be the sphere of radius r centered at point x. We will use the following notion of sphere packing [7, 14] in our algorithm.

Definition 3.1 (β-Packing) *Let β a positive real constant. Let S be a set of spheres and $P \subset \Omega$ be the centers of these spheres. Then S is a β-packing with respect to a spacing function f if*

- *For each point p of P, $B(p, f(p)/2) \in S$;*
- *The interiors of any two spheres s_1 and s_2 in S do not overlap; and*
- *For each point $q \in \Omega$, there is a sphere in S that overlaps with $B(q, \beta f(q)/2)$.*

To construct the mesh points for M', we first use the following procedures to generate a β-packing of Ω with respect to f by using as many mesh points from M as possible. Here β is a constant to be given later. M' is then the Delaunay triangulation of the centers of the β-packing.

Algorithm `Functional-Refining-Functional-Coarsening`

1. Let $S_1 = \{B(p, f(p)/2) | p \in M\}$;
2. For each element $t = \triangle q_1 q_2 q_3$ in M, let q be the mesh point q_i with the smallest $f(q_i)$ for $i = 1, 2$ or 3. Let b_t be the smallest box that contains t. We divide b_t into a set of uniform cells with the side length $c_5 f(q)/(2\sqrt{2})$, where c_5 is a constant given in Lemma 3.4. See Figure 3. Choose a random point in every cell that intersects t for a nonempty area, and for each such a point x, define a sphere with center x and radius $f(x)/2$. Let S_2 be the set of these spheres.

   In practice, we can use $(f(q) + ||x - q||)/2$ to approximate the radius $f(x)/2$. Note that $c_5 f(q) \leq f(x) \leq f(q) + ||x - q|| \leq c_3 f(q) \leq c_3/c_5 f(x)$. In other words, $(f(q) + ||x - q||)/f(x)$ is bounded both from above and below by constants. Hence, it is reasonable to use $f(q) + ||x - q||$ to approximate $f(x)$;
3. Let $S' = S_1 \cup S_2$;
4. Order the spheres in S' as the following: first, all spheres whose centers are on the boundary followed by all other spheres in S_1 in the order of increasing radii, and then followed by all spheres in S_2 in the order of increasing radii;
5. We say two spheres s_1 and s_2 in S' are *conflicting* if their interiors overlap. The conflicting relation defines a *Conflict Graph* (CG) over S'. Let S be the set of spheres which form the *Lexical-First Maximal Independent Set* ($LFMIS$) of CG;
6. Let M' be the mesh defined by the constraint Delaunay Triangulation of centers of S.

The lexical-first maxiaml independent set is defined as the following. The initial LFMIS is empty. Then we add a sphere with the smallest index that does not conflict with any spheres of the current LFMIS until no sphere can be added. The intuition is that we try to conform the boundary, and to use as many spheres from as possible. In addition, the smaller sphere has higher priority to be chosen, which will intuitively improve the quality of the new mesh M'. If we choose the larger spheres first, it is more likely that it will generate some big gaps among the selected spheres, which will deteriorate the quality of the new mesh.

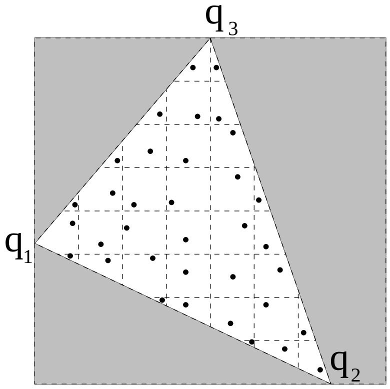


Figure 3: Sampling points in a triangle element.

3.3 Complexity of FRFC

To analyze the complexity of the *FRFC* algorithm, we first show that the number of the mesh points in the new mesh is linearly related to that of the original mesh.

Lemma 3.5 (Number of Sample Points) *The number of sampled points in an element $\triangle q_1 q_2 q_3$ is bounded by $2\sqrt{2}/(c_5 l)$.*

Proof: The number of cells generated in the triangle is no more than $2\sqrt{2} nn_M(q_i)/(c_5 f(q_i))$. The lemma follows from Lemma 3.1, i.e., $nn_M(q_i) \leq f(q_i)/l$. $\square$

The most time consuming part of *FRFC* will be the steps for computing the conflict graph and for computing the LFMIS of the conflict graph. The sphere-separator based divide-and-conquer algorithm [6] constructs the conflict graph in $O(|M| \log |M|)$ time if the spheres $S_1 \cup S_2$ satisfy the *constant ply* property. Let Γ be a collection of spheres. The *ply* of a point x, denoted by $ply(x, \Gamma)$, is the number of the spheres in Γ that contains x.

Lemma 3.6 (Constant Ply) *For $S' = S_1 \cup S_2$, there exists a constant c_6 such that for any point $x \in \Omega$,*

$$ply(x, S') \leq c_6. \tag{7}$$

Proof: Let $C(x)$ be the set of the centers of the spheres in S' that contain x. For any point $c \in C(x)$, we have $2||x - c|| \leq f(c)$. Note that $f(c) \leq f(x) + ||x - c||$ because f is 1-Lipschitz. Hence, $||x - c|| \leq f(x)$, i.e., point c is in $B(x, f(x))$.

Let $T(x)$ be a set of triangles that contain a point $c \in C(x)$. It is sufficient to show that the size of $T(x)$ is bounded by a constant due to that the number of sampled points in each triangle is bounded by a constant (See lemma 3.5). Note that the number of triangles is linearly related to the number of the mesh points because the mesh is a planar graph. Hence, we need only to prove that the number of mesh points of triangles $T(x)$ is bounded by a constant.

We first prove that, for any point $c \in C(x)$, $f(c) \geq 2/3 f(x)$. Note that $||c - x|| \leq f(x)$. There are two cases for the location of point c. If $||c - x|| \leq 1/3 f(x)$, then the fact that $f(x) - f(c) \leq ||c - x||$

implies $f(c) \geq 2/3f(x)$. If $||c - x|| > 1/3f(x)$, then the fact that $||c - x|| \leq 1/2f(c)$ implies that $f(c) > 2/3f(x)$. Hence, in both cases we have

$$f(c) > 2/3f(x). \tag{8}$$

Let $\triangle q_1q_2q_3$ be the element in $T(x)$ containing c. Let q be the point among q_1,q_2,q_3 that has the smallest spacing value. Lemmas 3.1 and 3.2 imply that $nn_M(q_i) \geq f(q_i)/L \geq f(c)/(c_3L)$. We have

$$nn_M(q_i) \geq 2f(x)/(3c_3L), \quad for\ i = 1, 2, 3. \tag{9}$$

From Lemmas 3.1, 3.2, and 3.4 we have $nn_M(q_i) \leq f(q_i)/l \leq c_3f(q)/l \leq c_3f(c)/(c_5l)$. Hence,

$$nn_M(q_i) \leq 2c_3f(x)/(3c_5l), \quad for\ i = 1, 2, 3. \tag{10}$$

It follows from Lemma 2.1 that $||c - q_i|| \leq el_M(q_i) \leq nn_M(q_i)/c_1$, which implies that, for every mesh point q_i of $T(x)$, $||x - q_i|| \leq ||x - c|| + ||c - q_i|| \leq f(x) + nn_M(q_i)/c_1$. Hence, for each mesh point q_i of $T(x)$,

$$||x - q_i|| \leq (3c_5l + 2c_3)f(x)/(3c_5l), \tag{11}$$

implying that the sphere $B(x, (3c_5l+2c_3)f(x)/(3c_5l))$ contains all the triangles from $T(x)$. Note that the spheres $B(q_i, nn_M(q_i)/2)$ are pairwise non-overlapping, and they are contained in the sphere $B(x, (c_5l + c_3)f(x)/(c_5l))$, because $nn_M(q_i) \leq 2c_3f(x)/(3c_5l)$, and q_i is contained by $B(x, (3c_5l + 2c_3)f(x)/(3c_5l))$.

By a volume argument, the number of mesh points in $T(x)$ is bounded by a constant R^2/r^2, where $R = 1 + c_3/(c_5l)$, $r = 2/(3c_3L)$. The lemma then follows from the fact that the number of points in each triangle is bounded by a constant $2\sqrt{2}/(c_5l)$. Note that the constant c_6 satisfies $c_6 \leq (2\sqrt{2}/(c_5l))(3R^2/r^2 - 6)$. □

We now analyze the time complexity of the algorithm. Because M is well-shaped, it has a linear number of elements and edges in terms of the number of mesh points $|M|$ [8]. Therefore, nn_M can be evaluated in $O(|M|)$ time. The time to compute the global spacing function f is $O(|M| \log(|M|))$. Notice that the mesh point p_j that defines $f(p)$ has the property that for each mesh point $p_k \in M$:

$$\delta(p_j)nn_M(p_j) + ||p_j - p|| \leq \delta(p_k)nn_M(p_k) + ||p_k - p||. \tag{12}$$

That is, p is contained in the additively weighted Voronoi cell of p_j. Fortune [3] shows how to apply the sweep-line technique to compute the additively weighted Voronoi diagram in $O(|M| \log(|M|))$ time.

The time complexity of step 1 of the algorithm `FRFC` is $O(|M| \log(|M|))$. During step 2, the number of the points sampled at any element $\triangle q_1q_2q_3$ is bounded by $2\sqrt{2}nn_M(q_i)/(c_5f(q_i))$. Lemma 3.1 implies that the number of sampled points is at most $2\sqrt{2}/(c_5l)$. Hence, the time complexity of step 2 is also $O(|M| \log(|M|))$. Note that the time complexity is $O(|M|)$, if we use $(f(q)+||x-q||)/2$ to approximate $f(x)$. The time to sort all the spheres during step 4 is $O(|M| \log(|M|))$. Because the ply of $S_1 \cup S_2$ is bounded by a constant (Lemma 3.6), we can apply the sphere-separator based divide-and-conquer algorithm [6] to construct the conflict graph in $O(|M| \log |M|)$ time. In addition, we know that the conflict graph has at most $O(|M|)$ number of edges. Computing the lexical first maximal-independent-set of the conflict graph then takes $\Omega(|M|)$ time. The Delaunay triangulation takes $O(|M'|\log(|M'|))$ time, where $|M'|$ is linear in $|M|$, because the total number of the sampled points is linear in $|M|$. Thus, we have the following theorem.

Theorem 3.7 (Complexity) *The time complexity of FRFC is $O(|M|\log(|M|))$.*

4 Quality and Size

We now show that `FRFC` returns a mesh M' that is well-shaped, and is of a size that is within a constant factor of the optimal possible.

4.1 Quality of the Mesh

We will use the following structure theorem of Miller, Talmor, and Teng [7] which states an equivalence relationship between β-sphere packing and well-shaped meshes to prove the quality of the generated mesh.

Theorem 4.1 (Sphere Packing and Well-Shaped Meshes)

1. *For any positive constant β, there exists a constant α depending only on β such that if f is a spacing function of Lipschitz constant 1 over a domain Ω and S is a β-sphere packing with respect to f, then the Delaunay triangulation M of the centers of S is an α well-shaped mesh; in addition, for each point p in Ω, $nn_M(p) = \Theta(f(p))$, where the constant in Θ depends only on β.*

2. *For any positive constant α, there exists a constant β depending only on α such that if M is an α well-shaped mesh, then the set of spheres*

$$S = \{B(p, nn_M(p)/2) : \text{for all mesh point } p \in M\},$$

is a β-packing with respect to $nn_M/2$.

We use the following lemma from Miller *et al* [5] to prove that $FRFC$ generates a β-packing sphere set.

Lemma 4.2 *Let P be a set of points in domain Ω. Let g be an γ-Lipschitz function defined on Ω. Let $S = \{B(p, g(p)) | p \in P\}$. If for any point $x \in \Omega$, $B(x, g(x))$ contains at least one point from P, then a LFMIS of the conflict graph of S is $(3+\gamma)/(1-\gamma)$-packing.*

Hence, we only need to prove that the sampling method of $FRFC$ guarantees that for any point $x \in \Omega$, $B(x, f(x)/2)$ contains at least one point from $S_1 \cup S_2$.

Lemma 4.3 (Dense Sample) *For any point x in the domain, the sphere $B(x, f(x)/2)$ contains at least one point from $S_1 \cup S_2$.*

Proof: It is sufficient to show that $B(x, f(x)/2)$ contains at least one cell generated during the sampling procedure. Let $t = \triangle q_1 q_2 q_3$ be the element that contains x. Let q be a mesh point q_i with the smallest $f(q_i)$, $i = 1, 2, 3$. The side length of the cell generated during the sampling procedure is $c_5 f(q)/(2\sqrt{2})$, where c_5 is given in Lemma 3.4. If the radius of the sphere is at least the diagonal of the cell, i.e., $f(x)/2 \geq c_5 f(q)/2$, then the sphere will contain a cell. By Lemma 3.4 that $f(x) \geq c_5 f(q)$. □

Noting that the spacing function $f/2$ used for spheres is 1/2-Lipschitz, we have the following theorem.

Theorem 4.4 *The S returned by the* FRFC *algorithm is a 7-packing with respect to f.*

Therefore, there exists a constant α such that the mesh M returned by the algorithm is α-well-shaped.

4.2 Size of the Mesh

We show that the new pacing function is good in the sense that it is the maximum function that satisfies the evolving meshing problem.

Lemma 4.5 (Maximality) *Let h be any spacing function of Lipschitz constant 1 over the domain Ω that satisfies the condition $h(p) \leq \delta(p) nn_M(p)$ for each mesh point p in M. Then for any point x in Ω, not necessarily a mesh point of M, $h(x) \leq f(x)$.*

Proof: Let $h_p(x) = h(p) + ||x - p||$. The fact that $h(p) \leq \delta(p) nn_M(p)$ implies that $h_p(x) \leq f_p(x)$. Let $h'(x) = min_{p \in M} h_p(x)$. Note that $f(x) = \min_{p \in M} f_p(x)$, and hence $h'(x) \leq f(x),\ for\ all\ x$. Now assume p_x is the point that drives x to get the smallest value for h', i.e., $h'(x) = h_{p_x}(x) = h(p_x) + ||p_x - x||$. Note that $h(x) \leq h(p_x) + ||p_x - x||$ because h is 1-Lipschitz function. Then, $\forall x, h(x) \leq h'(x) \leq f(x)$. □

Therefore, let M'' be any mesh that satisfies the condition of the Evolving Mesh Problem. We have for any point q in Ω, $nn_{M''}(q) \leq f(q)$, because $nn_{M''}$ is 1-Lipschitz function, and $nn_{M''}(p) \leq \delta(p) nn_M(p)$. The size optimality follows from the fact that f is a maximum spacing function that satisfies the condition of the Evolving Mesh Problem, (see Lemma 4.5), and the following lemma of [7].

Lemma 4.6 (Size of a Well-shaped Mesh [11]) *If M is an α-well-shaped mesh of n elements, then*

$$n = \Theta\left(\int_\Omega \frac{dA}{nn_M^2}\right). \tag{13}$$

First, by a simple volume argument, the number of the spheres in S is bounded by

$$\Theta\left(\int_\Omega \frac{dA}{f^2}\right), \tag{14}$$

Because f is point-wise larger than the nn function of any well-shaped mesh that satisfies the Evolving Mesh Problem. It follows that size of M' is within a constant factor of the best possible.

Theorem 4.7 (Main) `FRFC` *constructs a well-shaped mesh that satisfies the spacing condition given by δ. In addition, its size is optimal within a constant factor.*

The key to our algorithms in maintaining the well-shaped condition is to make sure that the shape condition does not deteriorate from M to M'. This is why we add new sampling points to regions near C-points to ensure the constant β in β-packing is maintained. Miller *et al* [7] showed that in their coarsening algorithm that no new point is needed for coarsening. However, to do so, they need to use the original finest mesh directly to generate the coarsening mesh at each level. In other words, if the original mesh is M_0, then mesh point of M_0 are used to build the conflict graph to generate the mesh points for M_i. If they simply use mesh points of M_{i-1}, then mathematically, they can not guarantee that the mesh points of M_{i-1} are dense enough for the well-shaped condition through the quality of the packing for M_i. In EMP, because of the mixed refinement, the original mesh does no longer provide fine enough sample points to guarantee the packing condition. Hence, new points has to be added. One of our objectives here is to add as small number of new points as possible, and meanwhile, by using as simple procedures as possible. In practice, for the moving boundary problem, there is no need to add new sample points to the back region of the moving boundary. We can use the algorithm of Miller *et al* [7] to coarsen the back region.

5 Practical Variations

The δ values decompose the mesh M into a collection of components of maximal submeshes where the δ values of all mesh points in each submesh are either larger than 1 (type C-submeshes), or smaller than 1 (type R-submeshes). In practice the number of such submeshes is bounded by a small constant. For example, this number in most problems with a moving boundary is 2 (one for the front-end of the moving boundary and one for the back-end). As observed in Section 3, we need to insert Steiner points in the submeshes that are required for refinement. From submeshes to be coarsened, we often need to remove some original mesh points. A practical variation of our scheme is to first refine the R-submesh by any adaptive refinement algorithm, such as quad/octree refinement and Delaunay refinement. Then we apply the one-level coarsening algorithm of Miller, Talmor, and Teng [7]. We now present a detailed procedure for the case where Delaunay refinement is used. Recall that the standard Delaunay refinement procedure contains three rules [11, 12]:

1. splitting boundary subsegment whose diametral sphere contains a mesh point other than its end-points in its interior by adding a Steiner point at its midpoint;

2. splitting a boundary subfacet whose equatorial sphere contains a non-coplanar mesh point by adding a Steiner point at its circum-center. However, if the new point would cause any subsegment of the subfacets to split, apply rule 1 to these subsegments instead.

3. splitting any simplex that does not satisfies the well-shape condition by adding a Steiner point at its circum-center. However, if the addition of this circum-center would cause any subsegment or subfacet to split, then apply rules 1 and/or 2 instead.

We add a fourth rule, which states as: splitting any simplex in which the current nn-spacing of any one of its mesh points is more than its δ-value times its initial nn-spacing by adding a Steiner point at its circum-center. However, if the addition of this circum-center would cause any subsegment or subfacet to split, then apply rules 1 and/or 2 instead.

Algorithm `Delaunay-Refining-Functional-Coarsening`
Input: A well-shaped mesh M and a list of positive reals δ.

1. Apply rules 1, 2, 3, 4 until all constraints on the spacing and shape at each mesh point are satisfied. Call the resulting mesh M_I.
2. Apply the one-level coarsening method of Miller, Talmor, and Teng [7] to M_I with coarsening factors given in δ to M_I to construct M'.

The following theorem follows directly from the main theorem of Ruppert [11] for 2D and of Shewchuk [12] for 3D and the coarsening result of Miller *et al* [7].

Theorem 5.1 `Delaunay-Refining-Function-Coarsening` *(DRFC) constructs a well-shaped mesh that satisfies the spacing condition given by* δ.

One of the shortcomings of `DRFC` is that it may construct a mesh that is larger than necessary. The reason is that in the refinement, we did not remove any original mesh point. In `FRFC`, we may replace some original mesh points in the R-submeshes by new Steiner points, which potentially reduce the mesh size. However, `DRFC` in general is more efficient.

When the lower bound l on δ is very small, the number of points introduced in each triangle could be very large, although it is a constant. This is undesirable, especially in the coarsening regions, (e.g., back-end of a moving boundary). In practice, we have a few alternatives:

- Do not add any points to triangles all of whose mesh points are C-points.
- Add only the barrycenter and/or midpoints of the edges rather than generating random quasi-uniform points based on the local grid.

Talmor [14] showed in her thesis that in practice no new point is needed for the region to be coarsened repeatly. We conduct experiments to verify this point in the context of EMP.

6 Experimental Results

The proposed algorithm is implemented and tested on a dynamic moving boundary problem. This section discusses the implementation of the algorithm and the performance of it.

We perform our program on a simple moving boundary problem, in which a circle, that stands as the moving boundary, is growing from almost the center of a box-shaped domain. See Figure 4. The motion of the circle is discretized so that it will reach the boundary of the domain in certain number of steps, that is related with the speed of the moving boundary. As the accuracy of the numerical simulation depends on it , it is crucial to have smaller mesh elements at the regions closer to the boundary. On the other hand to speed up the simulation one can use a coarser decomposition of the domain at the regions further away from the boundary. Hence, as the circle grows, regions outside the circle (the front end of the boundary) should be refined and the interior of the circle (the back end of the moving boundary) should be coarsened. At this point, we should note the use of an important parameter which specifies the width of the moving boundary i.e., the thickness of the moving circle. So the points that are on the boundary are assigned the lowest δ values. The refinement and the coarsening factors (δ values) for the other mesh points are computed as a function based on the distance to the moving boundary. For the experiment shown in Figure 4, δ values are linearly related with the distance to the boundary.

Figure 4 presents a six step simulation of the growing circle. Each row in the figure presents one iteration of the simulation. The first column shows the new spacing function represented by the

iteration	# of points	# of original points	% of original points	# of triangles	minangle
1	1416	1084	76.55	2702	17.29
2	1836	1328	72.33	3542	14.31
3	2183	1487	68.12	4236	14.86
4	2068	1469	71.03	4006	16.30
5	1625	1198	73.72	3117	10.23
6	1220	878	71.97	2308	14.36

Table 1: The performance numbers of the simulation shown in Figure 4

.

spheres, i.e., the spacing function value at a point is the diameter of the spehere centered at that point. The second column illustrates the spehere packing computed as the maximal independent set of the collection of the spheres in the first column. Finally, the third column shows the mesh as the Delaunay Triangulation of the centers of the spheres in the second column.

The Table 1 summarizes the performance of the implementation for the corresponding steps. Before the existence of the moving boundary there is an initial mesh of 1096 points. As soon as we introduce the growing circle, the number of points increases as expected since we refine the mesh around the moving boundary. However, the increase is at a reasonable level. As the circle meets the boundary of the domain, the number of mesh points decreases as expected. It is observed that the size of the mesh stays at an acceptable level throughout the simulation. Another equally important observation is that the percentage of the number of original mesh points is considerably high. This can be crucial for most simulation problems and can be achieved simply by giving the priority to the original points in the computation of the lexical first maximal independent set. In fact, we order the spheres by the type of the sphere and the size of the sphere. There are five different types of spheres with the following preference order: *corner spheres, boundary spheres, original spheres, Steiner spheres, and sampled spheres.* The corner spheres are those that must be selected to conform the mesh to the boundary of the domain. The boundary spheres are those spheres centered at the boundary of the domain. These spheres differ from the corner spheres in that they are not required to be selected to obtain a boundary conformal mesh. They are introduced for quality purpose. The original spheres are those centered at the points from the original mesh. The spheres centered at the sampled points of a previous iteration are called Steiner spheres. Those spheres centered at the sampled points of the current iteration are called sampled spheres.

We also observed that mesh quality does not deteriorate much during the simulation. In the worst case, the minimum angle of the mesh decreases to a level of 10.23 degrees. We observe that the smallest angle often comes from the region around the boundary. To improve the quality of the new mesh, we can borrow some ideas from the Delaunay refinement: when some points to be selected are too close to the boundary, we can split the boundary segment instead.

Notice that the growing circle example is a simplified scenario. Real-world numerical problems with a moving boundary are usually much more complicated. For these problems, one can use a set of points to approximate the boundary and move the points accordingly to simulate the motion. The refinement/coarsening factor of a point in the domain depends on its distance to the closest point in the set that approximates the moving boundary. Notice that the set of points used to simulate the moving boundary are changed also because of the density of the simulated points. Intuitively, the simulated points are distributed evenly on the simulated boundary.

7 Conclusion

In this paper, we present a unified approach for coarsening and refining evolving meshes We present some experimental results to show the effectiveness of our algorithm and its practical variations.. One application and motivation of our work is for solving time-dependent problems with a moving boundary. In our future work, we will explore the structure of the moving boundary and level sets to speed up the coarsening and refinement procedure. We will also work on incorporate our algorithm into some standard mesh generation software. In addition, all of the lemmas and theorems are applied to three dimensions if the radius-edge ratio is used as shape criterion of the well-shaped mesh. However, this does not prohibit the existence of slivers.

In the context of parallel implementation of the Evolving Mesh Problem, the need of mesh evolution could introduce load imbalance among processors, where the load measures the amount of work required by solving the Evolving Mesh Problem itself as well as by numerical calculations thereafter. We need to develop a mesh distribution estimation algorithm to incorporate with the dynamic load balancing scheme developed in [4].

References

[1] Babuška, I.; Aziz, A.K. (1976) On the angle condition in the finite element method. SIAM J. Numer. Anal., 13(2):214-226.

[2] Bern, M.; Eppstein, D.; Gilbert, J. R. (1990) Provably good mesh generation. In 31^{st} Annual Symposium on Foundations of Computer Science, IEEE, 231-241.

[3] Fortune, S. (1987) A sweep line algorithm for Voronoi diagrams. Algorithmica, 2:153-174.

[4] Li, X. Y.; Teng, S. H. (1998) Dynamic load balancing for parallel adaptive mesh refinement. In 5^{th} International Symposium on Solving Irregularly Structured Problems in Parallel, Berkeley, 144-155.

[5] Miller, G. L.; Talmor, D.; Teng, S. H. (1998) Data Generation for Geometric Algorithms on Non-Uniform Distributions. International Journal of Computational Geometry and Applications, accepted and to appear.

[6] Miller, G. L.; Teng, S. H.; Thurston, W.; Vavasis, S. A. (1993) Automatic mesh partitioning. In George, A.; Gilbert, J.; Liu, J., editors, Sparse Matrix Computations: Graph Theory Issues and Algorithms, IMA Volumes in Mathematics and its Applications. Springer-Verlag, 57-84.

[7] Miller, G. L.; Talmor, D.; Teng, S. H. (1997) Optimal Good Aspect Ratio Coarsening for Unstructured Meshes. In 8^{th} Annual ACM-SIAM Symposium on Discrete Algorithms, 538-547.

[8] Miller, G. L.; Talmor, D.; Teng, S. H.; Walkington, N. (1995) A Delaunay based numerical method for three dimensions: generation, formulation, and partition. In Proc. 27^{th} Annu. ACM Sympos. Theory Comput., 683-692.

[9] Miller, G. L.; Talmor, D.; Teng, S. H.; Walkington, N. (1998) On the radius–edge condition in the control volume method. SIAM J. on Numerical Analysis. ?? pages

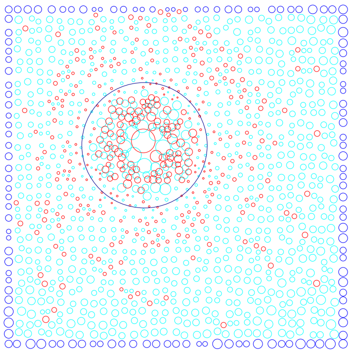
[10] Mitchell, S. A.; Vavasis, S. A. (1992) Quality mesh generation in three dimensions. Proc. ACM Symposium on Computational Geometry, 212-221.

[11] Ruppert, J. (1992) A new and simple algorithm for quality 2-dimensional mesh generation. In Third Annual ACM-SIAM Symposium on Discrete Algorithms, 83-92.

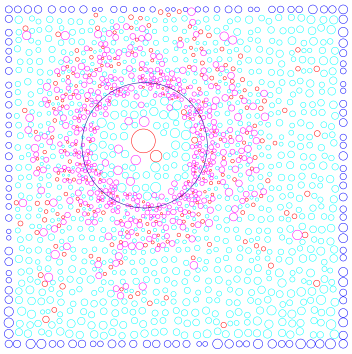
[12] Shewchuk, J. R. (1998) Tetrahedral mesh generation by Delaunay refinement. In 14^{th} Annual ACM Symposium on Computational Geometry, 86-95.

[13] Strang, G.; Fix, G. J. (1973) An Analysis of the Finite Element Method, Prentice-Hall.

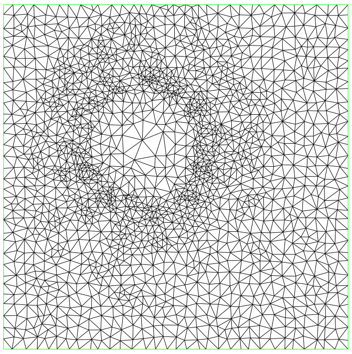
[14] Talmor, D. (1997) Well-Spaced Points for Numerical Methods. Ph.D thesis, Carnegie Mellon.



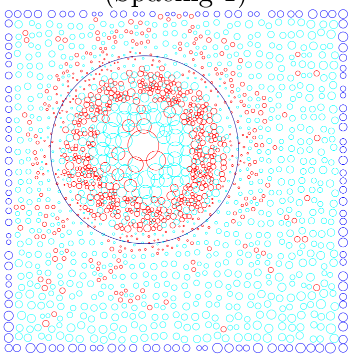
(Spacing-1)



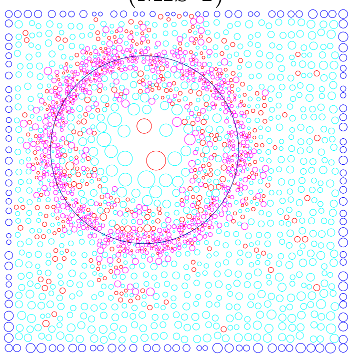
(MIS-1)



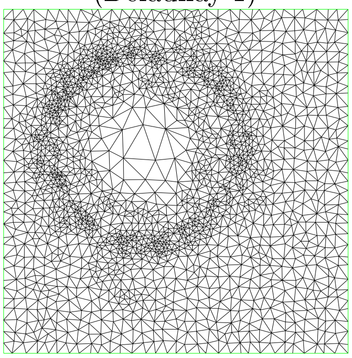
(Delaunay-1)



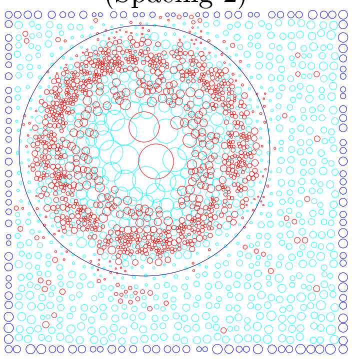
(Spacing-2)



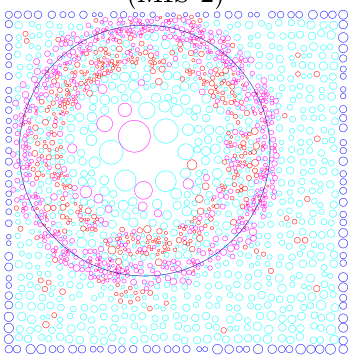
(MIS-2)



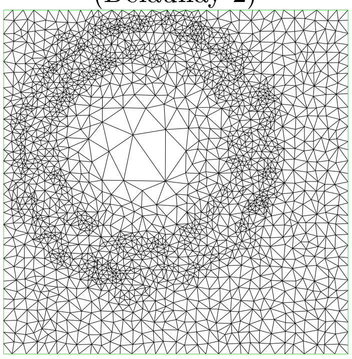
(Delaunay-2)



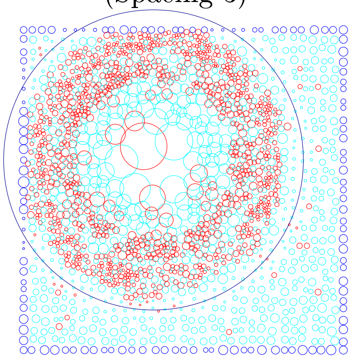
(Spacing-3)



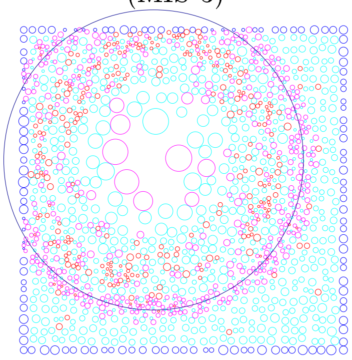
(MIS-3)



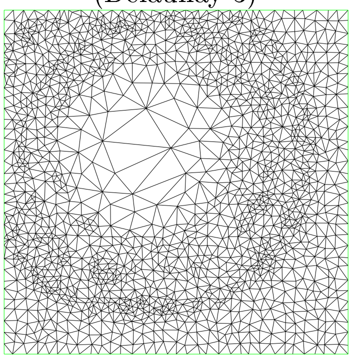
(Delaunay-3)



(Spacing-4)



(MIS-4)



(Delaunay-4)

Figure 4: Four iterations of the *FCFR* scheme.