William & Mary
CHARTERED 1693

W&M ScholarWorks

Dissertations, Theses, and Masters Projects

Theses, Dissertations, & Master Projects

2003

A Parallel Two Dimensional Delaunay Decoupling Method

Leonidas Linardakis
College of William & Mary - Arts & Sciences

Follow this and additional works at: https://scholarworks.wm.edu/etd

 Part of the Computer Sciences Commons

Recommended Citation

Linardakis, Leonidas, "A Parallel Two Dimensional Delaunay Decoupling Method" (2003). *Dissertations, Theses, and Masters Projects.* Paper 1539626828.
https://dx.doi.org/doi:10.21220/s2-m58d-n221

This Thesis is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

A Parallel Two Dimensional Delaunay Decoupling Method

A Thesis

Presented to

The Faculty of the Department of Computer Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Master of Science

---

by

Leonidas Linardakis

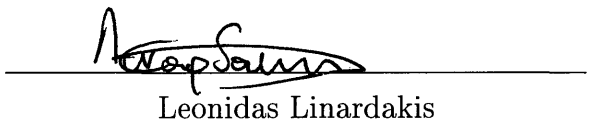
2003

APPROVAL SHEET

This thesis is submitted in partial fulfillment of

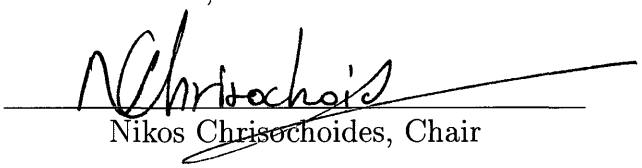
the requirements for the degree of

Master of Science



Leonidas Linardakis

Approved by the Committee, December 2003



Nikos Chrisochoides, Chair

Gianfranco Ciardo

Weizhen Mao

To Efi...

Table of Contents

ACKNOWLEDGMENTS

I'm deeply grateful to my advisor Nikos Chrisochoides for introducing and guiding me to the exciting field of the parallel mesh generation. Through the intricate discussions we had I've become a richer person. I would also like to thank the members of the committee Gianfranco Ciardo and Weizhen Mao for their help and their advice.

I was fortunate to have good friends and exceptional teachers. Whatever knowledge I have I owe it to them. I would like to thank especially Apostolos Thoma, as well as Theodoros Vidalis, who provided me the scientific and ethical foundations, while I was a student in the University of Ioannina. All the faculty members of the CS department in the College of William and Mary have instructed me in wisdom to various areas of Computer Science. Paul Stockmeyer revealed to me the playful nature of Mathematics, while Weizhen Mao showed to me the simplicity of the Complexity.

The administrative director of the department, Vanessa Godwin, was an invaluable guide to the mysterious vicinity of applications, forms and various deadlines.

Finally, I'd like to thank my colleagues, Kevin Barker, for his help in the parallel implementation of the MADD and for his insight of the parallel world, and Aaron Hawkins, for our explorations of the infinite book. Although we will never be able to finish it, just glancing through the pages is a pleasant and rewarding activity.

This work was partially supported by NSF ACI-0085969 and NSF NGS-0203974.

ABSTRACT

Creating in parallel guaranteed quality large unstructured meshes is a challenging problem. Parallel mesh generation procedures decompose the original mesh generation problem into smaller subproblems that can be solved in parallel. The subproblems can be treated as either completely or partially coupled, or they can be treated as completely decoupled. In this thesis we present a parallel guaranteed quality Delaunay method for 2-dimensional domains which is based on the complete decoupling of the subproblems. As a result the method eliminates the communication and the synchronization during the meshing of the subproblems. Moreover, it achieves 100% code re-use of existing, fine-tuned and well tested, sequential mesh generators. The approach we describe in this thesis presents for the first time an effective way to create in parallel guaranteed quality meshes with billions of elements in few hundreds of seconds, and at the same time demonstrates that these meshes can be generated in a efficient and scalable way. Our performance data indicate superlinear speedups.

A Parallel Two Dimensional Delaunay Decoupling Method

Chapter 1

Introduction

1.1 Delaunay Triangulation and Mesh Generation

Delaunay triangulation was introduced by Delaunay [21] in 1934 and is a triangulation such that the circumcircle (the circumscribed circle) of every triangle is empty, that is it does not contain any other vertex of the triangulation (see Figure 1.1). This property is referred as the *empty circumcircle property.* The advantages of the Delaunay triangulation is that it demonstrates adaptivity to the geometry and maximizes the minimum angle of the triangulation [33].

In the Delaunay mesh generation, points are inserted in the triangulation in order to improve the quality of the mesh (see Figure 1.1). A triangle is considered "bad" when it contains a small angle, or equivalently when the *circumradius to shortest edge ratio* is large. Typically the circumcenter of a such a bad triangle is inserted and new mesh is produced by re-triangulating the vertices. In addition to improving the quality of the mesh in terms of the angles this refinement procedure is used to

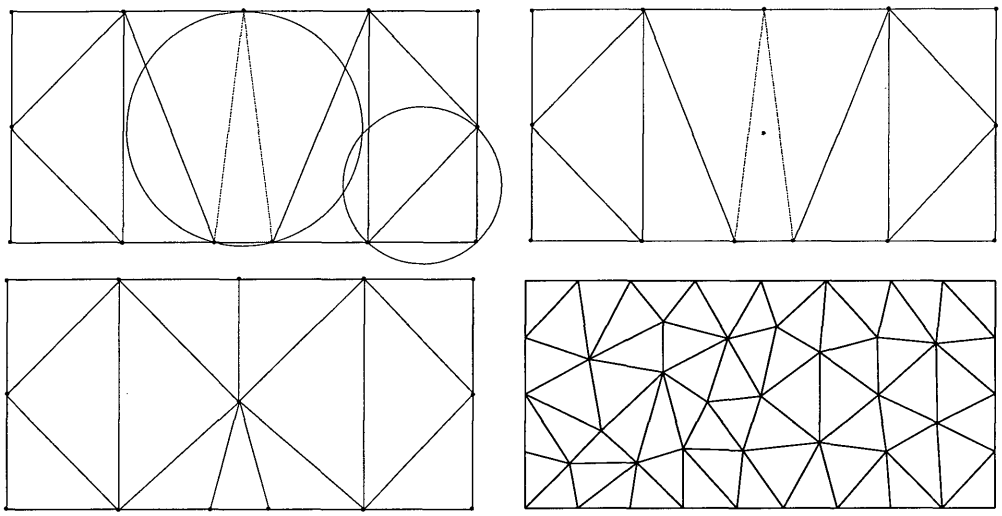


Figure 1.1: Delaunay triangulation and mesh generation. **Top left**, in Delaunay triangulation the circumcircles of the triangles are empty. **Top right**, in the Delaunay mesh generation the circumcenters of the 'bad' triangles are inserted and **down left** the mesh is re-triangulated. **Down right**, the mesh generation procedure is unpredictable and memory intensive.

decrease the size of the triangles, so that the maximum triangle area is bounded by a desirable size.

This procedure is not computational expensive, but is memory intensive and has unpredictable computational behavior, which is input dependent. In order to solve the problem of the memory intensive access. distributed memory machines can be used to create large meshes efficiently.

More information on Delaunay triangulation and mesh generation can be found in [23, 25, 45].

1.2 Parallel Delaunay Mesh Generation

Parallel mesh generation methods decompose the original meshing problem into smaller subproblems that can be solved (i.e., meshed) in parallel. The requirements for the parallel and distributed solution of the subproblems are: (1) *stability*, distributed meshes should retain the same level of quality of elements as the sequentially generated ones, (2) *efficiency*, and (3) *code re-use*, in order to leverage the ever evolving basic sequential meshing techniques and software.

In [20, 25] parallel mesh generation methods, for distributed memory computers or clusters of workstations (CoWs), are classified in terms of the *way* and the *order* the artificial boundary surfaces (interfaces) of the subproblems are meshed. Specifically, existing parallel methods are classified in three categories: (i) *A priori* methods, that first mesh (either in parallel [35], or sequentially [42]) the interfaces of the subproblems and then mesh in parallel the individual subproblems. (ii) *A posteriori* methods, that first solve the meshing problem in each of the subproblems in parallel, and then mesh the interfaces [20] so that the global mesh is conforming. (iii) *Simultaneous mesh generation and partitioning (SMGP)* methods, that simultaneously mesh and improve the quality of the interfaces[1] as they mesh the individual subproblems [19, 13, 17, 38].

In this thesis we present an a priori method that contributes in the state-of-the-art parallel mesh generation in the following three ways: (1) it guarantees the same level of quality of the mesh with the sequentially generated ones, (2) it eliminates

[1]The improvement of the interfaces is measured in terms of the surface to volume ratio.

communication and synchronization during the meshing of the subproblems, and achieves superlinear speedups with respect to the best (to our knowledge) sequential guaranteed quality mesh generator [44], and (3) achieves 100% code re-use, providing the ability to use the best sequential Delaunay mesh generators with no modifications. This is the first method (to the best of our knowledge) that eliminates communication and synchronization, and at the same time is based on a 100% code re-use of sequential codes. It is the only, so far, parallel guaranteed quality method that can achieve superlinear speedups, when compared to the best sequential mesh generation codes, and the first to create over 1B elements. The method can be used at the same time as sequential mesh generation, in order to create larger meshes in less time using one processor.

In [24] J. Galtier and P. L. George present a Parallel Projective Delaunay Meshing (P^2DM) method which guarantees the quality of the elements and eliminates communication and synchronization, but, depending on the geometry, it might suffer from setbacks which affect its efficiency. The setbacks are in the form of discarding completely the triangulation because the separators are not always Delaunay admissible as new points are inserted [24]. The problem of computing Delaunay admissible separators in the context of parallel Delaunay mesh refinement is solved in this thesis successfully for 2-dimensional domains.

A 2-dimensional Divide-and-Conquer Delaunay Triangulation (DCDT) algorithm and its parallel implementation are presented in [6]. The DCDT is based on finding

a Delaunay path, through a projection to a paraboloid, that separates the initial set in two equal sized subsets. Although this is an elegant and efficient procedure for Delaunay triangulation, it cannot be used for parallel mesh generation and refinement, which require new point insertion in the mesh, without significant extensions as the ones presented in [29], that introduce communication.

SMGP Parallel Guaranteed Quality Delaunay Mesh (PGQDM) generation methods appeared in [37] and in [18]. The PGQDM is communication intensive, and despite the fact that tolerates (masks) up to 90% of communication, its speedup is about 6 for 16 processors [18]. The second SMGP method, the Constrained Delaunay Mesh (PCDM) generation [13] is based on constrained Delaunay triangulation [10]. It reduces substantially the communication and eliminates synchronization, but still the speedup is 5.75 for 8 processors [13]. The PCDM implementation, as the PGQDM, does not re-use existing sequential Delaunay mesh generators, due to additional care for cavities that are constrained by internal boundary.

The method we present here requires high quality domain decompositions that (1) satisfy certain geometric constraints [45] regarding the angles, and (2) do not introduce significant constraints that will affect the efficiency of the mesh generator and the quality of the final mesh. In this thesis we propose a novel domain decomposition method for 2-dimensional geometries based on the medial axis of the domain. This method satisfies the above criteria, but it has the disadvantage of being difficult to extend to 3 dimensions.

In the rest of the thesis, we present in chapter 2 the medial axis domain decomposition method. In chapter 3 we proceed to decouple the mesh generation process of the individual subdomains, by defining and preprocessing a zone around the internal boundaries of the subdomains. In this chapter, contrary to past work [35, 24], we prove that the preprocessing of the zone completely decouples the subdomains. Finally, in chapter 4 we present the complete parallel mesh generation procedure, and in chapter 5 we provide experimental results that demonstrate the efficiency of our method.

Chapter 2

The Medial Axis Domain Decomposition Method

2.1 The Domain Decomposition Problem

Guaranteed quality mesh generation algorithms [11, 12, 41] and software [45, 44] generate elements with good aspect ratio and good angles. These algorithms require that the initial boundary angles are within certain good bounds. For example, Ruppert's algorithm [41] requires boundary angles (the angles formed by the boundary edges) no less than 60^o, in order to guarantee the termination. Since the separators are treated as external boundary, the domain decomposition should create separators that meet the requirements of the mesh generation algorithm. So, the constructed separator should form angles no less than a given bound Φ_o, which is determined by the sequential mesh generation procedure that will be used to mesh the individual subdomains.

The domain decomposition can be used in parallel mesh generation to explore data-parallelism, as in many other areas of scientific computing. The three fundamental issues in data-parallel computations are: *communication, synchronization and load balancing.* The parallel mesh generation method we propose eliminates communication and synchronization, using a proper decoupling (see Section 3.1) of the subdomains. However, we achieve the decoupling at the cost of some over-refinement, which is analogous to the size of the separators of the subdomains. Therefore, one of our objectives in the domain decomposition step is to minimize the size of the separators relatively to the area of the subdomains. Then the over-refinement we introduce is insignificant (see Section 5.3).

The third important issue that affects parallel program performance is the good balance of the work-load among the processors. The equidistribution of processors' work-load is achieved by over-decomposing [5] the domain, i.e. $N >> P$, where N is the number of subdomains and P is the number of processors. The created subdomains are distributed to the processors using an a priory estimation of the work-load, based on the area of the subdomains. This ab initio approach gives good results for uniform cluster environments (see Section 4.2). However, a dynamic load balancing approach can be adopted using general purpose runtime systems, like the ones presented in [2, 36], to migrate at runtime subdomains from overloaded processors to ones that completed their work. The area criterion for estimating the work load appears to be a good measure in the case of our method, for the following reason: the

decomposition procedure, as we will see, creates "good" angles and small separators, and the created subdomains tend to have similar shapes after the over-decomposition of the domain; since the geometries are similar, the work of the mesher is approximately proportional to the area of the subdomains. The above intuition is confirmed by the results in Section 4.2.

In summary, the domain decomposition criteria for parallel mesh generation are:

1. Create good angles, that is angles no less than a given tolerance Φ_o. The value of Φ_o is determined by the sequential, guaranteed quality, mesh generation algorithm.

2. The subdomains should have approximately equal size (area-wise).

3. The size of the separator should be relatively small i.e., minimize the ratio $\max\{|\mathcal{H}|/|\Omega_i|\}$, where $|\mathcal{H}|$ is the length of the separator and $|\Omega_i|$ is the area of the subdomains.

The first condition is essential, since it is the one that guarantees the termination of the mesh generation procedure and at the same time prevents the creation of new features, that will lower the quality of the final mesh. Criteria 2 and 3 are not required, but are desired for the efficiency of the parallel computations.

The domain decomposition that we propose here is independent from the decoupling procedure described in Section 3.1, and it can be used in other parallel mesh generation methods, like PCDM, that require good quality domain decompositions.

2.2 Medial Axis Domain Decomposition Method

The Medial Axis Domain Decomposition (MADD) method we propose is based on an approximation of the medial axis (MA) of the domain. The MA was introduced by Blum [7] as a way to depict the shape of an object, and has been studied extensively during the last two decades [9, 8, 14, 34, 43, 49]. In the context of mesh generation the medial axis has been used in [1, 23, 26, 40, 46]. The existing domain decomposition methods aim mostly to solve the load balancing problem and to minimize the communication [22, 30, 47]. For the first time in the parallel mesh generation the medial axis was proposed as a domain decomposition technique in [15].

One of the contributions of this thesis is that, in addition to the load-balancing goal, the MA is used to guarantee domain decompositions with separators which form good angles between them and the external boundary. Like existing methods our decomposition method also aims for separators whose size is small relatively to the areas of the subdomains.

In the rest of the thesis we define as a domain Ω the closure of an open connected bounded set in $\mathbf{R}^2$, and the boundary $\partial\Omega$ is defined by a planar straight line graph (PSLG), which forms a set of (non-intersecting) line segments connecting pairs of points. A circle $C \subseteq \Omega$ is said to be maximal in Ω, if there is no other circle $C' \subseteq \Omega$ such that $C \subsetneq C'$. The closure of the locus of the circumcenters of all maximal circles in Ω is called the *medial axis* Ω and will be denoted by $\mathrm{MA}(\Omega)$. The intersection of a boundary of Ω and a maximal circle C is not empty. The points $C \cap \partial\Omega$, where a

maximal circle C intersect the boundary, are called *contact points* of c, where c is the center of C. Every point $c \in \mathrm{MA}(\Omega) \setminus \partial\Omega$ has at least two contact points.

The domain decomposition method we propose is based on the following simple geometric property:

Lemma 2.1 *Let b a contact point of $c \in MA(\Omega)$. The angles formed by the segment cb and the tangent of the boundary of Ω at b are at least $\pi/2$.*

Proof: We prove the lemma in the general case when Ω has a piecewise C^1 boundary. Suppose that the proposition is not true. Then there is a point $c \in MA(\Omega)$ of the medial axis and a contact point $b \in \partial\Omega$ of c, such that cb forms an angle $\phi < \pi/2$ with the boundary at b. Take c to be the origin of the axes and cb to define the y axis. Without loss of generality we assume that ϕ is formed by the tangent from the right. Let $(x(s), y(s))$ be locally the normal parametric representation of the curve, with $b = (x(0), y(0)) = (0, y(0))$ and $x(s) \geq 0$. We have $y(0) > 0$. Since $\phi < \pi/2$, we have $y'(0) < 0$. Let $R(s) = x^2(s) + y^2(s)$ be the square of the distance between c and the points of the curve. Because b is a contact point of c, it must be $R(s) \geq R(0) = |cb|^2$. We have $R'(0) = 2y(0)y'(0) < 0$. This means that locally $R(s) < R(0)$, which is a contradiction. ■

The medial axis of Ω can be approximated by Voronoi points of a discretization of the domain [9, 8]. Our strategy is to make use of the property of Lemma 2.1, and to construct separators that consist of linear segments which connect the Voronoi points to the boundary. The approximation of the $MA(\Omega)$ is achieved in two steps:

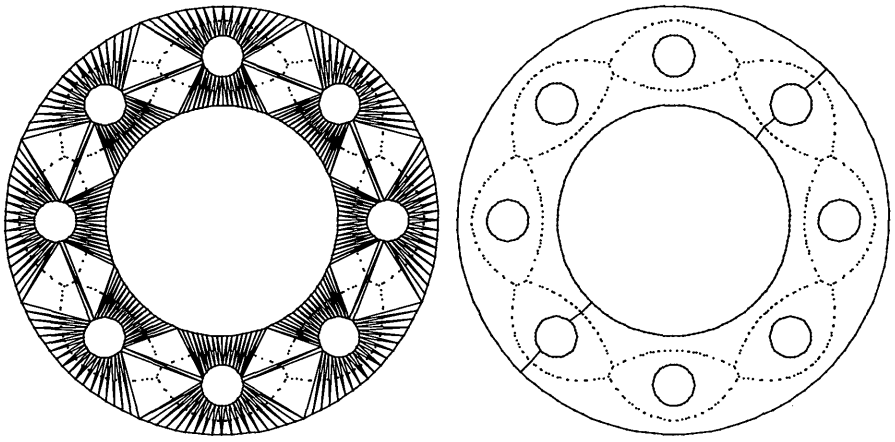


Figure 2.1: **Left:** The Delaunay triangulation of the pipe intersection. The circumcenters of the triangles approximate the medial axis. **Right:** The circumcenters are the Voronoi points. The separator is formed by selecting a subset of the Voronoi points and connecting them with the boundary.

(1) discretization of the boundary, and (2) computation of a boundary conforming Delaunay triangulation using the points from step (1). The circumcenters of the Delaunay triangles are the Voronoi points of the boundary vertices. The separators will be formed by connecting these circumcenters to the vertices of the Delaunay triangles. Figure 2.1 depicts the boundary conforming mesh of the cross section of a rocket (left), and the media axis approximation and a 2-way separator for the same geometry (right).

The level of the discretization of the boundary determines the quality of the approximation of the medial axis. However, our goal is not to approximate accurately the medial axis, but to obtain good angles from the separator. Therefore our criteria for the discretization of the domain will be specified by the quality of the angles. We achieve our goal by defining a new set of triangles:

Definition 2.1 *Let $\mathcal{D}$ be a Delaunay triangulation of a discretization D of the boundary $\partial\Omega$. We call a triangle $t \in \mathcal{D}$ a* junction triangle *if:*

1. *it includes its circumcenter c,*
2. *at least two of its edges are not in D,*
3. *at least two of the segments defined by the circumcenter and the vertices of t form angles $\geq \Phi_o$, both with the boundary and each other.*

The first criterion is set only for the simplicity of the MADD algorithm (see Section 2.2.1.2), in order to avoid negative weights and guarantee that at least two angles between the segments are good. The second prevents a decomposition that will create very small subdomains. The third criterion guarantees the quality of the angles. Let $a_1a_2a_3$ be the vertices of t. Then the third criterion demands the existence of at least one pair of segments a_ica_j, where c is the circumcenter of $a_1a_2a_3$, so that all the angles formed with these segments are greater or equal to Φ_o. Such pairs a_ica_j are called *partial separators* and they will be the candidates to form a complete separator. A complete separator decomposes a domain into two connected subdomains.

Let m be the number of holes of Ω. The level of refinement D we require for $\partial\Omega$ has to satisfy two conditions:

(i) In the Delaunay triangulation $\mathcal{D}$ of D there are at least $m+1$ junction triangles.

(ii) Every segment on the boundary D has an empty diametral circle.

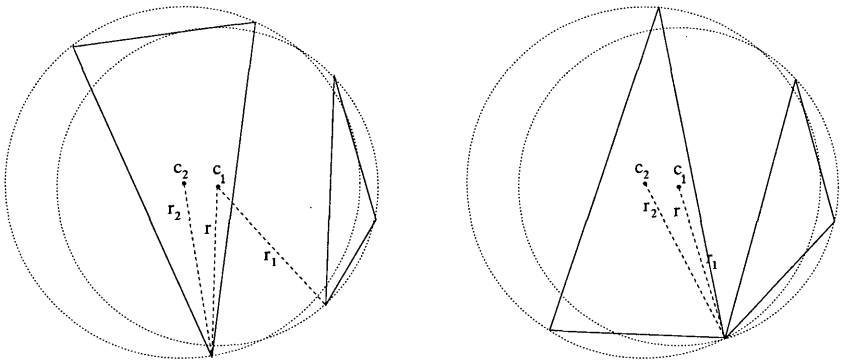


Figure 2.2:

The first condition in Definition 2.1 requires the existence of at least $m+1$ junction triangles. This ensures, as we will see in Section 2.2.2, that there is at least one complete separator formed by partial separators. The second condition guarantees that all the segments of D will appear as edges in $\mathcal{D}$. It also guarantees that all the circumcenters of the triangles of $\mathcal{D}$ are contained in Ω [45]. This in turn guarantees the existence of at least one triangle that includes its circumcenter. In order to prove it we will use the following Lemma:

Lemma 2.2 *Let A_1, A_2 be two triangles of a Delaunay triangulation, such that the circumcenter c_1 of A_1 is in the triangle A_2 and they don't have the same circumcircle. Let c_2 be the circumcenter of A_2 and r_1, r_2 be the radii of the circumcircles of A_1 and A_2 respectively. Then we have $r_1 < r_2$.*

Proof: Let r be the smaller distance of c_1 from the vertices of A_2, see Figure 2.2. Then $r \geq r_1$. Then we have $r_2 > r$, and consequently $r_2 > r_1$. ■

Lemma 2.3 *If all segments in D have empty diametral circles, then there is at least one triangle in the Delaunay triangulation $\mathcal{D}$ of D that includes its circumcenter.*

Proof: We know that, when the boundary segments have empty diametral circles, all the circumcenters of the triangles of $\mathcal{D}$ are in $\mathcal{D}$ [45]. We assume that the points are in general position, that is no four points belong to the same circle. We will prove the lemma by contradiction.

Suppose that the lemma is not true. Then for every triangle A_i there is another triangle $A_{i+1} \neq A_i$, such that the circumcenter c_i of A_i is included in A_{i+1}. Let r_i be the radius of the circumcircle of A_i. Since we assumed that no triangle includes its circumcenter, the sequence $< A_i >$ is infinite. On the other hand the set $\{t_i\}$ of all triangles in D is finite, so the sequence $< A_i >$ includes an element t_k twice. Then $A_l = A_m = t_k$, for some $l < k$. From the previous lemma we have $r_l < r_{l+1} < ... < r_m$, which contradicts to the fact that r_l and r_m are the radii of the same circle, and thus equal. So the lemma must hold. ■

The discretization of the boundary is determined by the number of the junction triangles that are created. As we increase the refinement, the Voronoi points approximate the points of the medial axis and the formed angles with the boundary tend be close to $\pi/2$. If we construct more junction triangles, and thus more partial separators, we have more choices to form a better separator, in terms of the quality of the angles, the size of the separator and the balance of the areas of the subdomains. In our experiments a rather small refinement (less than 700 additional points) gives

satisfying results. This of course depends on the geometry, and a way to predefine the refinement level of the boundary of the domain is a subject of further research.

2.2.1 The MADD Algorithm

The MADD algorithm uses as a starting point the approximation of the medial axis by the Delaunay triangulation $\mathcal{D}$, as described in the previous section. The complete separator is formed by partial separators i.e., segments inserted in junction triangles of $\mathcal{D}$; these segments connect the circumcenter of the triangles to two of their vertices. Figure 2.1 (right) depicts a complete separator for a 2-way decomposition of the pipe.

The partial separators connect two points of the boundary, since $\mathcal{D}$ is a boundary conforming triangulation. The properties of junction triangles permit the construction of good angles between the partial separators and the external boundary of the geometry. The MADD algorithm will select to insert a set of partial separators that will guarantee the decomposition of the domain into two subdomains. The selection of the partial separators is based on the minimization of the ratio of the size of the separators to the areas of the subdomains.

The MADD algorithm transforms the Delaunay triangulation $\mathcal{D}$ into a graph $G_{\mathcal{D}}$ which encapsulates the required information about the candidate partial separators. This information includes: (1) the topology of $\mathcal{D}$, which is used to guarantee that the inserted partial separators form a complete separator, and (2) the length of the partial separators and the area of the subdomains that will be created, which is used

to optimize the ratio of the length of separators to the subdomains area. After $G_{\mathcal{D}}$ is constructed, the graph is contracted, so that only the junction triangles of $\mathcal{D}$ are represented. Then the contracted graph is partitioned; the graph partitioning can be obtained by using any of the well known algorithms [32, 4, 27, 28, 30, 48], that decompose a connected graph into two connected subgraphs and minimize the ratio of the cut cost to the weights of the subgraphs. Finally the graph partition is translated into insertions of partial separators, which result into a 2-way decomposition. In summary the key steps of the algorithm are:

1. Create a graph $G_{\mathcal{D}}$ from the Delaunay triangulation $\mathcal{D}$.
2. Contract $G_{\mathcal{D}}$ into the graph $G'_{\mathcal{D}}$, so that only the partial separators in the junction triangles are represented as edges of $G'_{\mathcal{D}}$.
3. Partition the graph $G'_{\mathcal{D}}$, optimizing the cut-cost to subgraph weight ratio.
4. Translate the cuts of the previous partition into partial separators.

2.2.1.1 Construction of the Graph $G_{\mathcal{D}}$

In this step the Delaunay triangulation $\mathcal{D}$ is represented as a weighted graph, the dual graph of the edges of the triangles. Two nodes of the graph are adjacent if their corresponding edges belong in the same triangle. The length of the radius of the circumcircle of this triangle will be the weight of the graph edge. The weights of the

nodes are set to zero in this step, and they will be computed in the graph construction step (see Section 2.2.1.2).

Figure 2.3 (left) depicts the step for constructing the graph $G_{\mathcal{D}}$. One graph node is created for each edge of the triangulation, and two nodes are connected if they belong to the same triangle. Let d_{ij} be the node corresponding to the edge $a_i a_j$. The weight of the edge connecting d_{ij}, d_{jk} is the length $|c_l a_j|$, where c_l is the circumcenter of the triangle. For example, the edge that connects d_{12} and d_{25} has weight the length $|c_1 a_2|$. The above procedure is described by the following algorithm:

Algorithm 1 ().

1. **for** all the edges $a_i a_j$ in $\mathcal{D}$ **do**
2. Add node d_{ij} to the graph $G_{\mathcal{D}}$, with zero weight
3. **endfor**
4. **for** all triangles $t \in \mathcal{D}$ **do**
5. **for** the three pairs $(a_i a_j, a_j a_k)$ of edges of t **do**
6. Create a graph edge between the corresponding nodes d_{ij}, d_{jk},
7. with weight the length of the circumradius of t
8. **endfor**
9. **endfor**

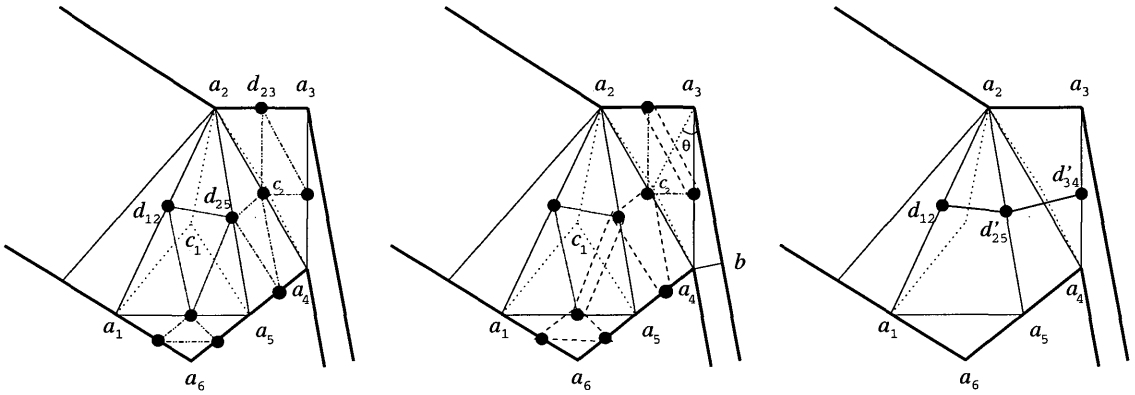


Figure 2.3: An example of creating the MADD graph. **Left** is a part of the Delaunay triangulation and the creation of the corresponding initial graph $G_{\mathcal{D}}$. **Middle**, the procedure of contracting the graph by combining the nodes of $G_{\mathcal{D}}$. The nodes connected by dashed lines are combined. **Right** is the final graph $G'_{\mathcal{D}}$ that corresponds to this part.

2.2.1.2 Graph Contraction

In this step the graph $G_{\mathcal{D}}$ produced from the previous step is contracted to a graph $G'_{\mathcal{D}}$, so that only the edges of junction triangles are represented as nodes in $G'_{\mathcal{D}}$. The nodes of $G_{\mathcal{D}}$ that correspond to edges of non junction triangles of $\mathcal{D}$ are contracted in $G'_{\mathcal{D}}$.

In order to contract the graph $G_{\mathcal{D}}$, first we iterate through all the triangles that are not junction triangles. The nodes of $G_{\mathcal{D}}$ that correspond to the three edges of a non-junction triangle are combined into a single node and the new node replaces the initial nodes in the external graph edges, while edges between the three initial nodes are deleted. The weight of the new node is the sum of the weights of the initial ones, plus the area of the triangle.

The remaining nodes correspond to the edges of junction triangles. Junction

triangles contain candidate partial separators, whose number may vary from one to three. From the three possible partial separators we keep the one that forms the greater minimum angle. Since in junction triangles there is at least one partial separator that forms angles no less than Φ_o, the selected partial separator forms angles $\geq \Phi_o$. We establish this partial separator by combining the two of the three nodes that correspond to edges of the triangle. Let $a_1a_2a_3$ be a junction triangle and c its circumcenter. Let d_{ij} be the corresponding node to the edge a_ia_j, then the weight of the node d_{ij} is updated by adding the weight of the area included by the triangle ca_ia_j. Let a_jca_k be the partial separator that forms the greater minimum angle. Then the nodes d_{ji} and d_{ki} are contracted into a single node, where a_i is the remaining vertex. The procedure is illustrated with the following example.

Example. Figure 2.3 (center) illustrates the procedure of contracting the graph. The bold lines indicate the external boundary. The triangles are part of the boundary conforming Delaunay triangulation of the domain. As above, we denote by d_{ij} the graph node that corresponds to the segment a_ia_j. We demonstrate four different cases.

Case I: The triangle $a_1a_5a_6$ has two edges on the boundary, so it is not a junction triangle, and the three corresponding nodes are combined to one. The edges connecting the new node d'_{15} are the external ones i.e., the edges that connect d_{15} to d_{12} and d_{15} to d_{25}. The weight of d'_{15} is equal to the area of the triangle $a_1a_5a_6$.

Case II: The triangle $a_2a_4a_5$ does not include its circumcenter and so it is not a

junction triangle. We follow the same procedure as above. The nodes d_{25}, d_{24}, d_{45} are contracted into a new node d'_{25}. The new node has weight the area of the triangle $a_2a_4a_5$ and is connected to the nodes $d_{12}, d'_{15}, d_{23}, d_{34}$.

Case III: The triangle $a_1a_2a_5$ is a junction triangle. The areas of the triangles formed by its circumcenter c_1 and its corners are added to the weight of the corresponding nodes. For example, the area $|a_2c_1a_1|$ is added to the node d_{12}, similarly the areas $|a_2a_5c_1|$, and $|a_1c_1a_5|$ are added to the nodes d'_{25} d'_{15}, respectively. Suppose that the partial separator $a_1c_1a_2$ is the one that that forms the greater minimum angle. Then the nodes d'_{15} and d'_{25} are contracted into a new node d'_{25} with its weight to be equal to the sum weights of the two previous nodes. The graph edge connecting the nodes d'_{15} and d'_{25} is deleted, while the two other graph edges are contracted into one edge connecting d'_{25} to d_{12}; the new edged weight is equal to the sum of the two previous edge weights, which is equal to the length of the partial separator $a_1c_1a_2$.

Case IV: The triangle $a_2a_3a_4$ is also a junction triangle. As for the previous triangle, first we add the areas of the triangles formed by the circumcenter c_2 and the vertices. The areas $|a_2c_2a_4|$, $|a_2c_2a_3|$, and $|a_3c_2a_4|$ are added to the weight of the nodes d'_{25}, d'_{23}, and d'_{34}, respectively. However, suppose in this case the angle θ, formed by the segment c_2a_3 and the external boundary segment a_3b, is less than Φ_o. Then the two partial separators that include this segment are rejected and we keep the separator $a_2c_2a_4$, which is the one that forms the greater minimum angle. The nodes d_{23} and d_{34} are be combined to the node d'_{34}. The new node is connected to

an edge with weight equal to the sum of the two previous edge weights, which is the length of the partial separator $a_2c_2a_4$. Figure 2.3 (right) shows the final graph.

The above procedure is described by the following algorithm:

Algorithm 2 ().

1. **for** all non junction triangles $t \in \mathcal{D}$ **do**
2. Combine the three nodes that correspond to the edges
3. of t, generating a new node d'
4. Add the area of t to the weight of d'
5. **endfor**
6. **for** all junction triangles $t \in \mathcal{D}$ **do**
7. Let c be circumcenter of t
8. **for** all edges a_ia_j of t **do**
9. Add the area of the triangle a_ica_j to the weight
10. the corresponding node d_{ij}
11. **endfor**
12. Find the partial separator a_ica_j in t forming a max min angle
13. Combine the nodes d_{ik} and d_{jk}, where a_k is the remaining vertex
14. **endfor**

2.2.1.3 The Construction of the Separator

After contracting the graph, the constructed graph $G'_{\mathcal{D}}$ is partitioned. The number of the edges of the graph is less or equal to the number of junction triangles, thus the size of the graph partitioning problem is significantly smaller than the element-wise dual graph of the boundary conforming Delaunay triangulation $\mathcal{D}$. Graph partitioning can be very expensive and has been an active area for several years [32, 4, 27, 28, 30, 48]. Any of the algorithms that give a partition of the graph into two connected subgraphs, with good cut cost to subgraph weight ratio, can be used as the graph partitioner for $G'_{\mathcal{D}}$. For algorithms that give non-connected subgraphs a check step must take place (see Section 5.2).

After partitioning $G'_{\mathcal{D}}$, the final step of the MADD is to construct the separator of the geometry. From the previous step we have a partition of the graph $G'_{\mathcal{D}}$ in two connected subgraphs. This partition will give a corresponding separator for the geometry. Each edge of the graph corresponds to a partial separator of the form $a_i c a_j$, where c is a circumcenter of a junction triangle and a_i, a_j are two of its vertices. For every graph edge that is cut by the partition we will insert the related partial separator in the geometry. In our example above (see Figure 2.4) the partial separator $a_2 c_2 a_4$ is created in the case that the graph partitioner chooses to cut the edge e_2.

The construction of the separator is described in the following algorithm.

Algorithm 3 ().

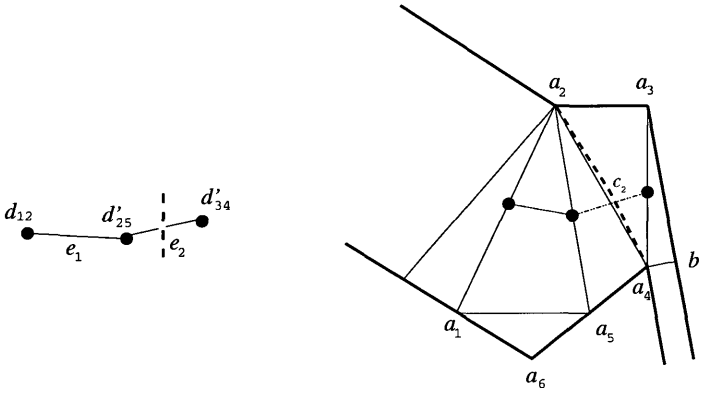


Figure 2.4: A partition of the graph and the corresponding separator, on the right, depicted with dashed lines.

1. **for** all triangles $t \in \mathcal{D}$ **do**
2. **if** one of the edges $a_i a_j$ of t belong to a different
3. subgraph from the other two edges **then**
4. Insert the partial separator $a_i c a_j$,
5. where c is the circumcenter of t
6. **endif**
7. **endfor**

The algorithm scans all the triangles and identifies those triangles whose edges correspond to nodes disconnected after the graph partition. In these triangles the partial separators are inserted, separating the edges that don't belong to the same subgraph. In Figure 2.4 the partial separator $a_2 c_2 a_4$ separates the edge $a_2 a_4$ from the edges $a_2 a_3$ and $a_3 a_4$. The set of all these inserted partial separators establishes a (complete) separator for the domain, as we will see in Section 2.2.2.

The ratio of the cost of the cut to the weight of the subgraphs is translated to the ratio of the total length of the separator to the area of the subdomains. Provided that the graph partitioner gives a good cut cost to subgraph weight ratio, the ratio of length of the separator to the area of the subdomains is also good. This way we obtain separators of relatively small size, and the areas of the subdomains are balanced. Moreover, since all the partial separators, by the construction of $G'_{\mathcal{D}}$, form good angles, the constructed separator forms good angles. In summary, the constructed separator meets the quality properties 1-3 we had required in the beginning of the section.

2.2.2 Proof of Correctness

In this section we prove that the MADD algorithm decomposes the domain Ω in two connected subdomains. We remind that the domain Ω is the closure of an open connected bounded set and the boundary $\partial\Omega$ is a PSLG that formed a set of linear segments which do not intersect. A separator $\mathcal{H} \subseteq \Omega$ is a finite set of simple paths (a continuous 1-1 map $h : [0,1] \to \Omega$) that do not intersect and define a decomposition A_1, A_2 of Ω in the following way: A_1 and A_2 are connected sets, with $A_1 \cup A_2 = \Omega$, and for every path $U \subset \Omega$ that connects a point of A_1 to a point of A_2, we have $U \cap \mathcal{H} \neq \emptyset$.

Lemma 2.4 *Let m be the number of holes in Ω and n the number of junction triangles. If $n > m$, then there is a separator for Ω formed by partial separators.*

Proof: We will prove the lemma by induction on m. If $m = 0$ then $n \geq 1$ and there is at least one partial separator. In this case, every partial separator is a separator for Ω, since every simple path $f : [a, b] \to \Omega$, with $f(a), f(b) \in \partial\Omega$ and $f(a, b) \subset \Omega^o$, is a separator for Ω.

Suppose the lemma is true for $m = q$, we will prove it is true for $m = q + 1$. We have that $n > q+1$. If for a partial separator acb, where $a, b \in \partial\Omega$, we have that both a, b don't belong to the boundary of a hole, then acb forms a separator, as in the case $m = 0$. In the case that one of the points a, b belong to the boundary of a hole O, then by inserting the partial separator acb we eliminate O. The new domain has q holes and $n-1 > q$ junction triangles, and can be decomposed by partial separators by the inductive hypothesis. Therefore there is a separator formed by partial separators. ■

Theorem 2.2 *Let m be the number of holes in Ω and n the number of junction triangles. If $n > m$, then the MADD decomposes Ω in two subdomains.*

Proof: Let $e_i, i = 1, ..., n$ be the edges of the contracted graph $G'_{\mathcal{D}}$ created by MADD. Each of these edges corresponds to a partial separator $h_i, i = 1, ..., n$. We will show that every decomposition of the graph $G'_{\mathcal{D}}$ corresponds to a decomposition of Ω formed by partial separators, and vice versa.

Let $E = \{e_i, i \in I\}$ be the set of edges that the graph partitioner cuts, creating two subgraphs G_1, G_2. Let $\mathcal{H} = \{h_i, i \in I\}$ be the set of partial separators that are correspond to these edges. Finally, let $A_1, A_2 \subset \Omega$ be the two corresponding areas to the subgraphs G_1, G_2. Obviously $A_1 \cup A_2 = \Omega$. From the construction of the

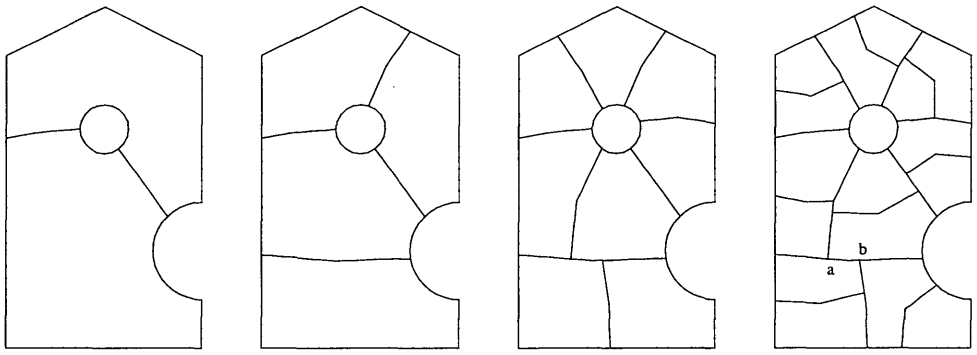


Figure 2.5: N-way partitions, where $N = 2, 4, 8, 16$, by the MADD divide and conquer method. Metis [31] was used as the graph partitioner and Triangle [44] produced the Delaunay triangulation.

graph we have that the connected subgraphs correspond to path connected areas of Ω. Assuming that the graph partitioner decomposes $G'_{\mathcal{D}}$ in two connected subgraphs, then G_1, G_2 are connected, and so A_1, A_2 are also connected. Every path $U \subset \Omega$ from a point of A_1 to a point of A_2 corresponds to a path U' in $G'_{\mathcal{D}}$ form a node of G_1 to a node of G_2. Since the edges E decompose G_1 from G_2, we have $U' \cap E \neq \emptyset$. Let $e_j \in U' \cap E$. Then we have $U \cap h_j \neq \emptyset$, and the path U intersects $\mathcal{H}$. Thus $\mathcal{H}$ is a separator for Ω. Working backwards we see that a separator for Ω corresponds to a partition of the graph. The existence of such a separator is proved in Lemma 2.4, and this completes the proof. ■

2.2.3 N-way Decomposition

So far we have described the MADD procedure for a 2-way decomposition. In the following section we will describe a decoupling procedure which is applied on multiple

subdomains and decouples the mesh generation procedure for all the given subdomains. In order to create more than two subdomains we can apply the MADD in a divide and conquer way (see Figure 2.5). When a 2-way separator is created, it is discretized and then every subdomain is decomposed independently. The resulting decomposition shows good adaptivity to the geometry. This approach requires to recalculate the Delaunay triangulation of the subdomains. We can do that by just inserting the segments of the separator in the existing triangulation. These segments should be refined, and possibly the edges of the boundary, so that the empty diametral circle property of the boundary, including the separators, is maintained.

Since every subdomain is decomposed independently, the discretization of the separators, which form the internal boundary, should be permanent. In practice, the size of the segments created by the discretization of the domain is much larger than the ones created by the mesh generation procedure. Here we should take into account that the level of decomposition is proportional to the size of the mesh we want to create. Thus, in the general case, the discretization does not create actual artificial constraints to the mesh. Figure 2.5 depicts that no new artifacts are introduced, given that segments like ab will be refined further.

In our method we refine even further the internal boundaries in order to decouple the subdomains, and our results show that the size and the quality of the mesh is not affected. For a more detailed experimental analysis see Section 5.3.

An advantage of the divide a conquer approach is that it is easy to be implemented

in parallel. In our implementation we have followed a parallel MADD divide and conquer strategy to create multiple subdomains. The method is described in detail in Section 4.1.

Chapter 3

The Decoupling Method

3.1 The Decoupling Zone

The separators and the subdomains created by the MADD procedure have good quality in terms of the shape and size. Our goal though is to be able to create Delaunay meshes independently for each subdomain, and the previous procedure cannot guarantee this. In order to create the mesh independently in each subdomain we have to ensure that the final mesh will be Delaunay conforming. Delaunay conformity, in the context of Delaunay triangulation, can be explored using a projective method [6]. A study of conditions for a priory conformity for constrained Delaunay triangulations is presented in [39]. A method for independent mesh generation in each subdomain using a projective separator is presented in [24], but it does not always guarantee a priory Delaunay conformity.

In order to ensure the Delaunay conformity in the mesh generation context we will refine the separators using conditions derived from the mesh refining algorithm. A

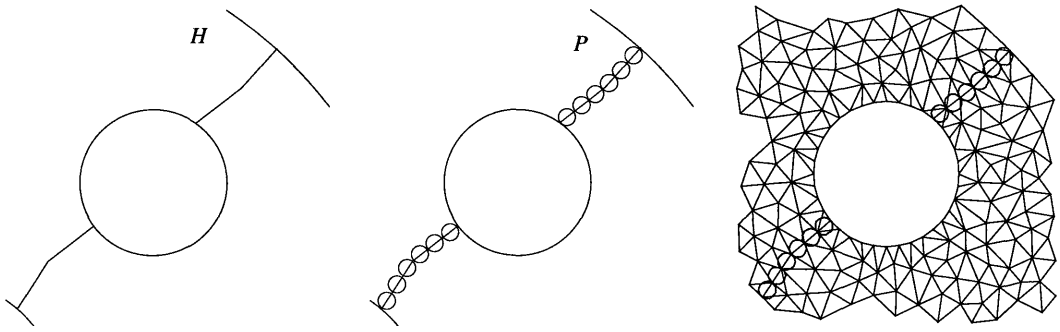


Figure 3.1: A fraction of the pipe intersection. **Left:** Part of the separators $\mathcal{H}$ inserted by MADD. **Middle:** Refining $\mathcal{H}$ gives a decoupling path $\mathcal{P}$; the decoupling zone $\mathcal{Z}_{\mathcal{P}}$ is depicted. **Right:** Ruppert's algorithm was applied on the subdomains with an element area restriction; $\mathcal{Z}_{\mathcal{P}}$ is empty and $\mathcal{P}$ is invariant. The final mesh is Delaunay conforming.

special "zone" around the segments of the separators (see Figure 3.1) will guarantee that the mesh generation procedure can be applied independently on each subdomain, giving a Delaunay conforming mesh for the whole domain, formed by the union of all the submeshes.

Let $\mathcal{M}$ be a Delaunay mesh generation procedure. Let $D = \partial\Omega$ be a PSLG, where Ω is a domain as described in the previous section. Let $\mathcal{P}$ be a set of piecewise linear separators that decompose the domain Ω in n subdomains Ω_i and let $D_i = \partial\Omega_i$ be the boundaries of the subdomains.

Definition 3.1 *The set of the open diametral circles of all the segments that form $\mathcal{P}$ is be called the* decoupling zone of $\mathcal{P}$ *and is denoted by $\mathcal{Z}_{\mathcal{P}}$.*

Definition 3.2 *$\mathcal{P}$ is a* decoupling path *with respect to $\mathcal{M}$, if after applying $\mathcal{M}$ independently on the subdomains Ω_i, $i = 1, ..., n$, the decoupling zone $\mathcal{Z}_{\mathcal{P}}$ is empty.*

Proposition 3.3 *Let M_i the mesh produced by $\mathcal{M}$ on the subdomain Ω_i. If $\mathcal{P}$ is a decoupling path with respect to $\mathcal{M}$, then the union $\cup M_i$ is a conforming Delaunay triangulation.*

Proof: Let M be the Delaunay triangulation of the vertices $V_M = \cup V_{M_i}$ of $\cup M_i$. We will prove that $M = \cup M_i$, by showing that the set of edges S of M are identical to the set of edges $\cup S_i$ of $\cup M_i$, thus the two triangulations are the same and $\cup M_i$ is a conforming Delaunay triangulation.

First we observe that $\mathcal{P}$ is a subset of both S and $\cup S_i$, because its decoupling zone is empty. For any edge $ab \in S$ there are two cases: (i) Both end points a, b belong to the same subdomain M_j, $a, b \in V_{M_j}$. (ii) $a \in M_i$ and $b \in M_j \setminus M_i$.

Case (i). Suppose $a, b \in V_{M_j}$. From the local Delaunay property, there is an empty circumcircle C of ab which does not include any points in V_M. Because $V_{M_j} \subseteq V_M$, C must be empty in the set V_{M_j}. Thus $ab \in S_j$ and $ab \in \cup S_i$.

Case (ii). We will show that this case cannot occur, there is no edge $ab \in S$ such that $a \in M_i$ and $b \in M_j \setminus M_i$. Suppose we have such an edge ab. Then $ab \subset D$ and, since the subdomains M_i and M_j are separated by $\mathcal{P}$, a and b are separated by $\mathcal{P}$. So $ab \cap \mathcal{P} \neq \emptyset$. On the other hand we have $\mathcal{P} \subseteq S$, which means that two edges of the triangulation M intersect. This contradicts the definition of a triangulation [23].

Since case (ii) cannot occur, we conclude from case (i) that $S \subseteq \cup S_i$. The two triangulations M and $\cup M_i$ must have the same number of edges, so we have $S = \cup S_i$, and thus $M = \cup M_i$. This proves the proposition. ■

Proposition 3.4 *If the algorithm $\mathcal{M}$ is a mesh refinement algorithm, then the decoupling path $\mathcal{P}$ is invariant during the steps of $\mathcal{M}$, in which the Delaunay property is maintained.*

Proof: Suppose that during the procedure $\mathcal{M}$ an edge $s \in \mathcal{P}$ is destroyed. That means that the diametral circle C_s of s includes some point. Since $\mathcal{M}$ does not remove points, C_s will not be empty after the termination of $\mathcal{M}$. This contradicts the definition of the decoupling path. ■

Proposition 3.3 proves that, provided that we have constructed a decoupling path, the subdomains can be meshed independently and the final mesh will be Delaunay conforming. Our next step will be to construct a decoupling path from the separators created by MADD.

The decoupling path is defined with respect to a mesh generation procedure and, in many cases [11, 41], the stopping conditions of the mesh generation algorithm allow us to compute the length of the edges of the separators, so that these edges will form a decoupling path. Then we only have to refine the segments of the separators, acquiring this predefined length.

3.2 Ruppert's algorithm

For the mesh procedure we will consider Ruppert's algorithm [41]. This is a mesh refinement algorithm for 2 dimensions that guarantees the quality of the elements. It creates an initial triangulation and follows an incremental approach to refine the

mesh. Triangles which have circumradius to shortest edge ratio greater than $\sqrt{2}$ are split, by inserting points in their circumcenters and constructing a new Delaunay triangulation. If a point to be inserted encroaches the diametral circle of a boundary edge, then, instead of inserting this point, the boundary edge is split in half. The algorithm maintains the Delaunay property after the insertion of each point. In order to guarantee the termination of this procedure the boundary angles should be at least 60^o.

Let D be a PSLG, as defined above. An entity is either a vertex or a segment of the boundary; two entities are incident when they share a common point. The *least feature size* of D is defined as the minimum distance between two non incident entities [45]; it will be denoted by $\text{lfs}_{\min}(D)$. The following proposition holds [45]:

Proposition 3.5 *Suppose that any any two incident segments of D are forming an angle no less than 60^o. Ruppert's algorithm terminates when applied on D, giving a mesh of triangles with circumradius to shortest edge ratio at most $\sqrt{2}$ and with no triangulation edge shorter than $lfs_{\min}(D)$.*

The only requirement for Ruppert's algorithm is that the boundary angles must be at least 60^o. Provided that our initial boundary D satisfies this criterion, we can apply MADD to decompose Ω using an angle bound $\Phi_o = 60^o$. So, both the constructed separators and the external boundaries form angles $\geq 60^o$. Consequently the created subdomains are acceptable for this mesh generation algorithm.

3.3 The construction of the decoupling path

Let $D = \partial\Omega$ be the boundary of the domain Ω, and $\mathcal{H}$ the set of separators in Ω created by the MADD method using an angle bound of $\Phi_o = 60^o$. Let $D_i = \partial\Omega_i$ be the boundaries of the created subdomains and $D_{\mathcal{H}} = D \cup \mathcal{H}$.

In order to construct a decoupling path $\mathcal{P}$ from the separators $\mathcal{H}$ we will refine $\mathcal{H}$ by inserting points along its edges, obtaining a desirable segment length. The calculation of this length is based on a parameter k. Let $L = \min\{|s|/\ s \text{ is a segment of } \mathcal{H}\}$. Let k be a real constant parameter, such that

$$0 < k \leq \min(\text{lfs}_{\min}(D_{\mathcal{H}}), L/4). \tag{3.1}$$

The parameter k will be calculated from the conditions of the algorithm, so that it can be guaranteed that no edge will be created with length less than k.

The following lemma describes the refining procedure of $\mathcal{H}$.

Lemma 3.1 *Let s be a segment of $\mathcal{H}$. Then there is $\nu \in \mathbf{N}$ such that, after inserting $\nu - 1$ points b_i on s, we have $\frac{2}{\sqrt{3}}k \leq |b_i b_{i+1}| < 2k$ for any two consequent points b_i, b_{i+1}.*

Proof: Let l be the length of the segment s and ν such that $2(\nu-1)k \leq l < 2\nu k$. Then, by dividing the s into ν equal subsegments, we have for the length l' of the subsegments: $\frac{2(\nu-1)}{\nu}k \leq l' < 2k$. For $\nu \geq 3$, we have $\frac{2(\nu-1)}{\nu} > \frac{2}{\sqrt{3}}$, and this proves the lemma. ■

Let $\mathcal{P}$ be the separators $\mathcal{H}$ after we have inserted the points b_i, as described in the previous lemma, and let $D_{\mathcal{P}} = D \cup \mathcal{P}$. The following Lemmas hold.

Lemma 3.2 *Let b_i, b_{i+1} two consequent points inserted on a segment s of $\mathcal{H}$. Then the diametral circle of $b_i b_{i+1}$ is empty.*

Proof: The diametral circle C of $b_i b_{i+1}$ is contained in the diametral circle of s, which by the MADD construction does not include any of the points of $D_{\mathcal{H}}$.

The remaining points to be examined are the inserted points b_j. We have that all the angles are greater than 60^o and, from Lemma 3.1, no created segment is less than half of any other created segment. Consequently, C cannot contain a point b_j created by the refining procedure. ■

Lemma 3.3 *The following inequality holds: $lfs_{\min}(D_{\mathcal{P}}) \geq k$.*

Proof: We have from the relation 3.1 that $\text{lfs}_{\min}(D_{\mathcal{H}}) \geq k$. We will examine the distances created by the inserted points.

Let b_i be a point inserted in a segment s of $\mathcal{H}$. For the distance d of b_i from a non incident to s segment we have $d \geq \text{lfs}_{\min}(D_{\mathcal{H}}) \geq k$. The same holds for the distance d' from points that are not incident to s, because we have $d' \geq d \geq k$.

For the distance d between b_i and an incident segment we have $d \geq \sin 60^o \cdot \frac{2}{\sqrt{3}} k = k$. Finally, the distance between b_i and a point that belongs to an incident segment is greater than the distance d of the previous relation, and this completes the proof.

■

The previous lemma demonstrates the property that will be used to prove that $\mathcal{P}$ is a decoupling path. Our next step will be to calculate the parameter k.

Ruppert's algorithm can be applied using either the quality criterion for the circumradius to shortest edge ratio, or by adding a criterion for the maximum area of the created elements. We will calculate k for this two cases separately. We will prove that $\mathcal{P}$ is a decoupling path for the two cases: (I) When Ruppert's algorithm is applied with only the quality criterion of the circumradius to shortest edge ratio. (II) When it is applied with an additional criterion for the maximum triangle area.

3.3.1 Case I: The ratio criterion

In this case we are only interested for the circumradius to shortest edge ratio. Since k gives a bound for the size of the created segments, we would like k to be as big as possible and at the same time satisfy the relation 3.1. Proposition 3.5 and Lemma 3.3 indicate that we can define $k = \min\{\text{lfs}_{\min}(D_{\mathcal{H}}), L/4\}$.

Proposition 3.6 *Define* $k = \min\{lfs_{\min}(D_{\mathcal{H}}), L/4\}$ *and let* $\mathcal{P}$ *be the piecewise linear separators as constructed in Lemma 3.1. Then* $\mathcal{P}$ *is a decoupling path with respect to Ruppert's algorithm.*

Proof: According to Proposition 3.5, Ruppert's algorithm when applied to a subdomain D_i, will not create segments less than $\text{lfs}_{\min}(D_i)$. We will show ad absurdo that the decoupling zone $\mathcal{Z}_{\mathcal{P}}$ is empty after the termination of the algorithm.

Suppose that $\mathcal{Z}_{\mathcal{P}}$ is not empty after the mesh procedure and some points have been inserted in it. That means that some boundary segments of $\mathcal{P}$ have been encroached and thus have been split in half. From Lemma 3.1 the length of the segments of $\mathcal{P}$ is less than $2k$ and by splitting them the created segments will have length less than k. This contradicts to Proposition 3.5 because, from Lemma 3.3, we have $\text{lfs}_{\min}(D_i) \geq \text{lfs}_{\min}(D_{\mathcal{P}}) \geq k$.

Thus the decoupling zone $\mathcal{Z}_{\mathcal{P}}$ is empty after applying Ruppert's algorithm, and $\mathcal{P}$ is a decoupling path with respect to this algorithm. ■

Corollary 3.7 *$\mathcal{P}$ remains invariant during Ruppert's algorithm execution.*

Proof: Ruppert's algorithm does not remove points and maintains the Delaunay property after inserting a point. The corollary is a direct consequence of the previous proposition and of Proposition 3.4. ■

Proposition 3.6 states that we can process the subdomains independently, using Ruppert's algorithm, and the final mesh will be Delaunay conforming and of guaranteed quality. Next we will examine the case where we have an additional condition for the area of the triangles.

3.3.2 Case II: The ratio and max area criteria

In this case, besides the circumradius to shortest edge ratio condition, we have an additional criterion for the maximum triangle area. In many cases we want to construct Delaunay meshes, not only with good quality of angles, but also of a desired

maximum size. Let A be a bound to the maximum triangle area, then all the triangles of the final mesh will have an area at most A. To achieve this, the mesh generation algorithm will split the triangles in two cases: (a) Because of the bad circumradius to shortest edge ratio. (b) Because the area of the triangle is greater than A.

We will calculate k so that the previous results will remain valid.

Lemma 3.4 *Let l be the smallest edge of a triangle with area greater than A and circumradius to shortest edge ratio at most $\sqrt{2}$. Then $l > \sqrt{\frac{A}{\sqrt{2}}}$.*

Proof: Let r be the circumradius of the triangle. Then $\frac{r}{l} \leq \sqrt{2}$ and $A < r \cdot l$. So, $A < r \cdot l \leq \frac{l^2}{\sqrt{2}} \Rightarrow l > \sqrt{\frac{A}{\sqrt{2}}}$. ■

We want to define k in such a way that the mesh generation procedure will not create edges smaller than k. The previous lemma indicates that we should have $k \leq \frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}}$. We will take $k = \min\{\text{lfs}_{\min}(D_{\mathcal{H}}),\ L/4,\ \frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}}\}$. Then Lemma 3.3 holds, and we have the following theorem:

Theorem 3.8 *Let $k = \min\{lfs_{\min}(D_{\mathcal{H}}),\ L/4,\ \frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}}\}$ be the parameter for the point insertion procedure in Lemma 3.1, and $\mathcal{P}$ the produced set of separators. Then $\mathcal{P}$ is a decoupling path with respect to Ruppert's algorithm with the criteria of maximum circumradius to shortest edge ratio $\sqrt{2}$ and maximum triangle area A.*

Proof: There are two cases for splitting a triangle: a) because of its circumradius to shortest edge ratio, or b) because of its area.

When Ruppert's algorithm splits a triangle because of its circumradius to shortest edge ratio it does not create edges smaller than $\text{lfs}_{\min}(D_{\mathcal{P}}) \geq k$. If a triangle is split because of its size, then from Lemma 3.4 we have that the smaller created edge will be no less than $\frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}} \geq k$. In both cases no edge smaller than k will be created.

It is easy to see now that the decoupling zone $\mathcal{Z}_{\mathcal{P}}$ will be empty, after Ruppert's algorithm has been applied on the subdomains D_i with the additional condition of a maximum triangle area A. If this was not so, then some edge of $\mathcal{P}$ would be encroached and split. From Lemma 3.1 the new edges will be smaller the k, which is a contradiction. ■

In summary, the procedure of preprocessing the separators created by MADD, as described in Lemma 3.1, creates a decoupling path with respect to Ruppert's algorithm in both cases of the quality and the size criteria. This will allow us to generate Delaunay meshes, independently for each subdomain, with good quality and of desired size. The final mesh, formed by the union of the submeshes, is Delaunay conforming. As a result, this procedure decouples the domain and allows us to parallelize the mesh generation procedure, while eliminating the communication between the processors.

Chapter 4

Parallel Guaranteed Quality Mesh Generation

4.1 The Parallel Delaunay Decoupling Procedure

The procedure for the parallel mesh generation consists of two steps:

1. The parallel MADD (PMADD) phase. In this step the domain is decomposed using the divide and conquer MADD method.

2. The mesh generation phase. This step is performed independently for each subdomain and includes two sub-steps:

   (a) The decoupling of the subdomains by refining the interfaces, as described in Section 3.3.

   (b) The mesh generation on the subdomains. In this step the sequential mesh generator is used as a library and is applied independently on each subdomain.

During the PMADD phase the domain is over-decomposed (i.e. we create $N >> P$ subdomains, where P is the number of processors), in order to achieve good load balancing (see Section 4.2). The PMADD method is implemented using a master/worker model. Processor 0 is used as the master processor, while all the processors, including processor 0, are used as worker processors. The master processor maintains a sorted list of the areas assigned to each processor. In each iteration of the PMADD procedure a decomposition request is sent from the master processor to the processors assigned with larger total areas. The processors that receive such requests decompose their larger subdomain in two subdomains using MADD. One of the two new subdomains is sent to a processor with small total assigned area. The procedure is repeated until all N subdomains are created.

The area of the subdomains is used to estimate the work-load for the mesh procedure (see Section 4.2). The goals of the PMADD is to minimize the larger area and to distribute the subdomains uniformly to the processors. Once the PMADD phase is finished no data movement takes place. This is an approximate criterion for the load balance, other means [2] for dynamic load balancing can be used.

After the requested number of subdomains have been created, the master processor sends requests to all processors to mesh the subdomains assigned to them. Each processor iterates through its subdomains and performs two steps:

(a) Meshing the interfaces, where the separators created by the MADD are refined using the decoupling procedure described in Section 3.3, according to the given mesh

quality criteria. The parameter k, that determines the refinement of the separators, is computed before the mesh generation phase begins, and is used to refine the internal boundaries of all the subdomains. The same orientation and the same procedure is used for each of the segments of the separators, establishing the conformity of the inserted points.

(b) The mesh generation procedure is applied on the subdomains independently. The sequential mesh generator is used as is, in the form of a library. As proved in Section 3.1, the created meshes are Delaunay conforming.

The procedure terminates when all the meshes for subdomains have been created. The parallel procedure is described next:

Algorithm 4 ().

1. **Master Processor:**
2. Read the definition of the domain Ω
3. Initialize and maintain a sorted list of the areas of the subdomains
4. **while** the current number of subdomains is less than N **do**
5. **send** decouple requests to processors that are assigned
6. large area of subdomains
7. **receive** replies about decoupling and area information
8. **endwhile**
9. **send** requests to processors to mesh their subdomains
10. **receive** replies **until** all processors completed meshing

```
11.    send requests for termination
12.
13.  Worker Processors:
14.    while not terminate do
15.        receive request from Master and/or other workers
16.        if request is to decouple then
17.            Apply MADD on the largest subdomain
18.            send reply to Master
19.            send a new subdomain to other processor
20.        endif
21.        if request is to receive a subdomain then
22.            Add the new subdomain to this worker's mesh-queue
23.            send reply to Master
24.        endif
25.        if request is to start meshing then
26.            for each assigned subdomain do
27.                Refine the separators according to the decouple procedure
28.                Apply the sequential mesh generator on the subdomain
29.            endfor
30.            send completion message to master
31.        endif
```

32. **endwhile**

During the PMADD phase, the first P subdomains are created in $\lg(P)$ iterations. The total number of iterations for the parallel MADD phase is $\frac{N-P}{P/2} + \lg(P) = 2(M-1) + \lg(P)$, where M is the average number of the final subdomains per processor. Typical values for M in our experiments vary between 12 and 20. The procedure is using in average $\frac{N-1}{2(M-1)+\lg(P)} = \frac{PM-1}{2(M-1)+\lg(P)}$ processors per iteration.

This divide and conquer approach is not optimal, but the cost is very small (see Section 5.4), with respect to the cost for the mesh generation. On the other hand it achieves a good load balance among the processors, which is a more significant factor for the total performance of the parallel mesh generation (see Section 5.4.2). In the next section we present in detail the load balance attained using the parallel MADD.

4.2 Load Balancing

Our experiments show that more than 99% of the total time is spent in the meshing phase (see Section 5.4), which does not suffer from communication or synchronization cost. Thus, the work-load balance among the processors is the main parameter that affects the performance of the method. The load balancing problem for mesh refinement is a difficult problem, because of the unpredictable computational behavior of the meshing procedure. The problem becomes more approachable by the use of the PMADD for over-decomposing the domain. The resulting subdomains have similar

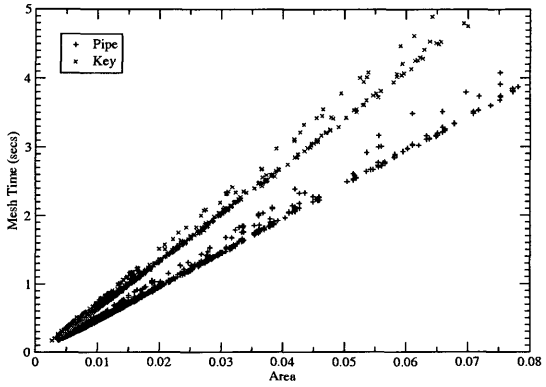


Figure 4.1: Mesh time for different sizes of subdomains of the key and the pipe geometry.

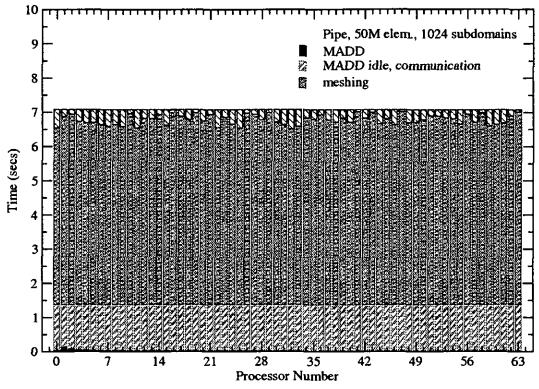


Figure 4.2: The work balance for 64 procs, 50M elem.

geometric shapes, and their area is proved to be a good measure for estimating the work load for the mesh generator.

Our experimental data show, for the geometries we tested so far, that the parallel MADD procedure creates subdomains with similar "good" shape (see Figure 5.1), when the number N of subdomains is large. Figure 4.1 shows that, as we increase N, and thus decrease the area of the subdomains, the meshing time converges, with very small differences between subdomains of similar size. This result demonstrates that the area of the subdomain can be used to estimate the work-load of the mesher for this subdomain. Of course this depends on the geometry of the original domain, which is one of the parameters that determine the level of required decomposition. An adaptive to the geometry approach for the PMADD would optimize the results, and this is a subject of future work.

The load balance among the processors is achieved by balancing the total area

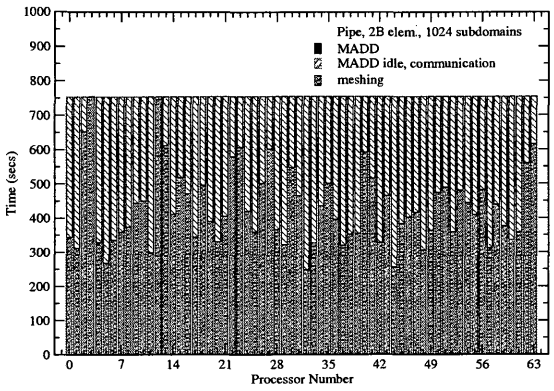


Figure 4.3: The work balance for 64 procs. 2B elem., 1024 subdomains for the pipe.

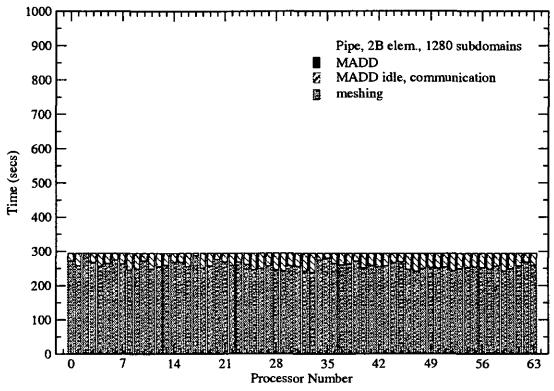


Figure 4.4: The work balance for 64 procs. 2B elem. 1280 subdomains for the pipe.

of the subdomains assigned to each processor. The first effort to create subdomains with similar sizes takes place during the graph partition. This result though is not guaranteed, and the obtained subdomains can have differences in size. By over-decomposing we have the ability to distribute the subdomains, so that each processor is assigned approximately the same total size. Moreover, the random distribution of the subdomains gives a more uniform assignment of subdomains that differ from the average in terms of size and geometry. The results of this simple approach are good. Figure 4.2 depicts the load balance among 64 processors for the pipe geometry, for 1024 subdomains and 50M mesh size. This picture is typical in most cases. However, we have observed that the load balance does not depend only on the geometry and the size of the subdomain, but also on size of the created mesh.

Figure 4.3 shows the load balance for the same decomposition of the pipe, as in Fig. 4.2, this time for a mesh size of 2 billion elements. We see that the good load

balance of the Figure 4.2 is destroyed. The reason for this is that the time for creating larger meshes is much more sensitive to area and geometry differences. The answer to this problem is to increase N. In this way we improve two parameters: *i*) the size of the mesh for each subdomain is decreased, and thus the time to create it is less sensitive to the differences, and *ii*) a more uniform assignment of the subdomains can be accomplished. Figure 4.4 shows the balance for the same mesh size, 2 billion elements, by decomposing it into 1280 subdomains. This small increase of the number of subdomains gives an impressive improvement, the load balance is satisfactory and the total time is decreased in less than half, the reasons are described in Sections 5.3, 5.4.1.

The previous example shows that the load balance is sensitive to the size of the final mesh. The level of the required decomposition depends not only on the geometry and the number of the processors, but mainly on the size of the final mesh. Let E be an estimation for the final size of the mesh in millions of elements. From our experiments we found that, for our setup, the number of subdomains should be at least $N = \frac{E}{1.6}$. This means that in average 1.6M elements will be created for each subdomain. A higher decomposition has, of course, higher time cost, but this cost is insignificant against the gain, Figures 4.3 and 4.4, as well as the results in the next section demonstrate it.

Chapter 5

Experimental Results

5.1 Performance Evaluation

We evaluate the Parallel Delaunay Decoupling (PDD) method with respect to three requirements listed in the Introduction: (1) stability, (2) parallel efficiency, and (3) code re-use. Our experimental data indicate that the PDD method is stable i.e., the elements of the distributed mesh retain the same good quality of angles as the elements generated by the Triangle (see Figures 5.3 and 5.8 (right)); at the same time it is very efficient as our fixed and scaled speedup data (see Figures 5.7, and 5.8 (left)) indicate. Finally it is based on 100% code re-use i.e., existing sequential libraries like Metis and Triangle are used without any modifications for the parallel mesh generation.

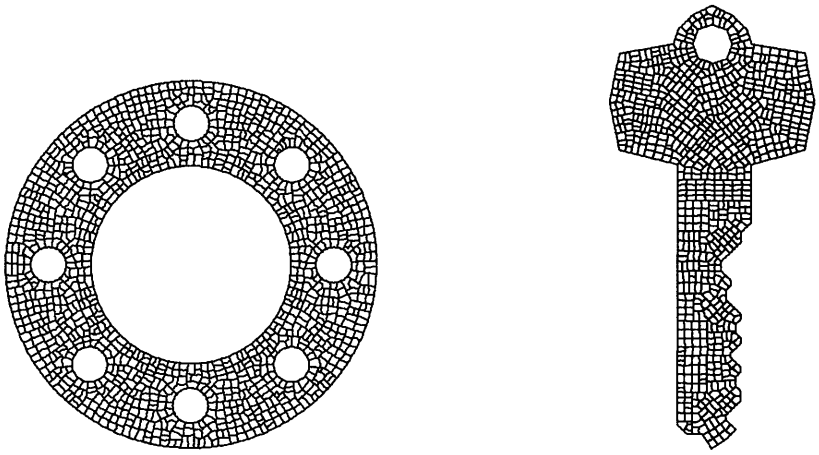


Figure 5.1: **Left:** The Pipe domain divided in 1200 subdomains. **Right:** The Key domain divided in 768 subdomains.

5.2 Experimental Setup

We have used two model domains (see Figure 5.1): The *Pipe*, a cross section of rocket from a NASA model problem where the peripheral pipes are used to cool the main cylinder in the center that contains combustion gases, and the *Key*, a domain provided with Triangle. We ran three sets of experiments: (1) to observe the the behavior of the MADD and Decoupling method in sequential execution for small meshes, 4-5 million (M) elements, (2) to calculate the fixed speedup for fixed size meshes of the order of 40-50M elements, and (3) to compute the scaled speedup for meshes whose size range from 12M to 2 billion (B) elements.

The programming language for our implementation was C++ and DMCS [3] was used as the communication substrate. The Triangle [44] library was used for the mesh generation procedure as well as for the creation of the Delaunay triangulation during the MADD procedure. The parameters passed to Triangle for the mesh generation

were two: (a) for the quality the elements (Ruppert's algorithm is used to achieve circumradius to shortest edge ration less then $\sqrt{2}$), and (b) for the maximum area of the generated elements. Also, Metis [31] was used for the graph partitioning step in the MADD procedure. The cases that Metis returned non-connected subgraphs were recognized and discarded. All the libraries where used without modifications, minimizing the cost for the parallel implementation and achieving 100% code-reuse.

All the experiments ran on SciClone, a high-performance computing environment in the College of William and Mary. SciClone is a heterogeneous cluster of Sun workstations which use Solaris 7 operating system. For our experiments we have used a subcluster of 32 dual-cpu Sun Ultra 60 workstations 360 MHz, with 512 MB memory and 18.2 GB local disk. Networking was provided by a 36-port 3Com Fast Ethernet switch (100Mb/sec).

5.3 Sequential Experiments

We ran a set of sequential experiments in order to compare the sequential Delaunay decoupling method, where we over-decompose the domain, with Triangle, the best known publicly available sequential guaranteed quality Delaunay mesh generation code for two dimensional domains. In these experiments we examine the affects of the decoupling procedure with respect to the performance of the mesh procedure, the size of the final mesh, which indicates that the over-refinement we introduce is insignificant, and the quality of the elements in terms of the angle distribution. The

size of the meshes we created is limited by the size (5.5M) we were able to generate with Triangle due to memory limitations. However, using the Delaunay decoupling method we were able to generate more than 30M on a single processor.

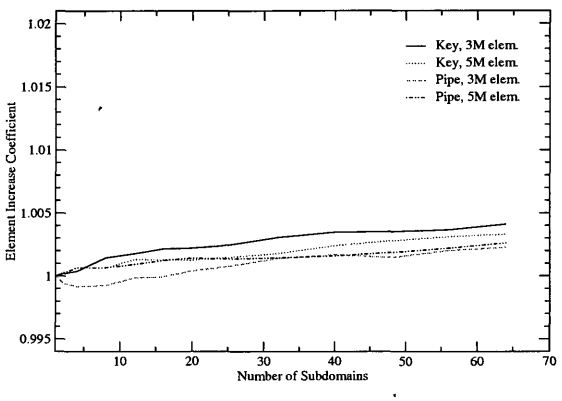


Figure 5.2: The increase of number of elements for decoupling into different number of subdomains.

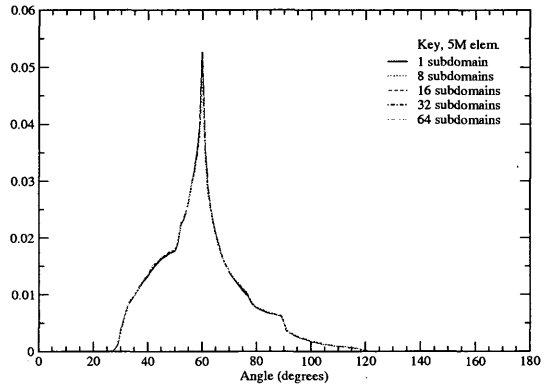


Figure 5.3: The angle distribution for different number of subdomains.

Figure 5.2 shows the ratio of the size of the decoupled meshes over the size of the non-decoupled mesh, which is a measure of the over-refinement we introduce when we decouple the domains.

Subdomains	1	8	16	32	48	64
Key elements	5,193,719	5,197,066	5,200,395	5,203,023	5,208,215	5,210,857
Total time	46.146	38.414	38.204	37.590	37.322	37.333
Pipe elements	5,598,983	5,602,668	5,605,819	5,607,055	5,609,404	5,613,624
Total time	59.263	41.342	41.046	40.370	40.352	40.147

Table 5.1: The number of elements and the total time (in seconds) for the same mesh generation parameters and for different levels of decoupling. The times do not include the mesh merging procedure.

Similarly, Table 5.1 presents the number of elements for different levels of decou-

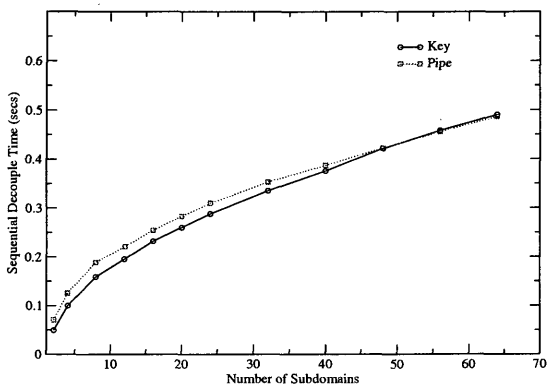


Figure 5.4: The time for the sequential MADD.

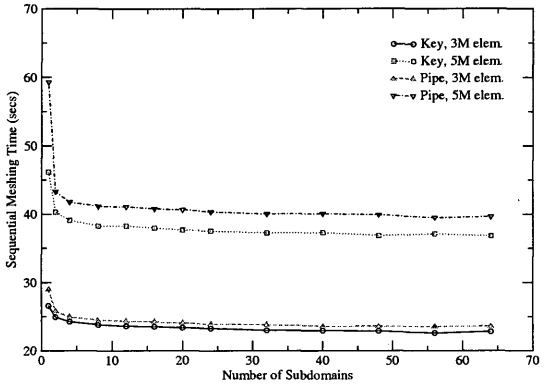


Figure 5.5: The time for sequential meshing after decoupling into subdomains. The times do not include the mesh merging procedure.

pling. The over-refinement is insignificant, it is less than 0.4%, despite the intense over-decompostion (less than 90K elements per subdomain).

The overhead of the sequential MADD method is approximately linear with respect to the number of subdomains, see Figure 5.4. This overhead is small compared to the mesh generation time. The total execution time using the sequential decoupling procedure is decreased up to 68% of the time it takes for Triangle to generate a mesh with the same quality. As the size of the mesh increases the performance of the decoupling procedure compared to Triangle is improving even further, because the size of the working set for each subdomain is smaller and the Delaunay mesh algorithm used in Triangle has a non-linear time complexity [44].

The quality of the elements produced after the decoupling of the domain into subdomains is evaluated by comparing the distribution of angles. We compare the angles of the elements from both the non-decoupled mesh generated by Triangle and

the decoupled ones generated by our method. Figure 5.3 shows that the distribution is the same. The above results hold as we scale the mesh size in our parallel experiments.

In summary, the decoupling method demonstrates merits even for sequential mesh generation. The gains in the performance from the better memory utilization cover the small overheads due to decoupling and over-refinement, while the element quality is independent of the decoupling, which shows that our method is stable regarding the quality of the mesh.

5.4 Parallel Experiments

We performed two sets of experiments in order to calculate the fixed and scaled speedup using 8, 16, 32, and 64 processors. With 64 processors we were able to generate 2.1 billion (B) high quality elements for the Pipe in less than 3.5 minutes, while using Triangle [44] on a single workstation we were able to generate 5.5 million (M) elements in about one minute (see Tables 5.1 and 5.3).

In the rest of the section we present performance data for both the parallel medial axis domain decomposition (PMADD) method and the parallel mesh generation. The PMADD procedure is evaluated in terms of its total parallel execution time which includes some communication and idle time and the maximum computation time spend on a single processor. The parallel mesh generation phase does not require communication and its performance is measured in terms of maximum and average computation time of processors. The ratio of these two numbers is used to measure

the load imbalance of the parallel meshing phase.

Finally, we evaluate the scalability of the method in terms of two performance criteria: (1) the **average time** that it takes for one element to be created on a single processor, over all the processors and elements that are created, and (2) the **overhead** cost (due to decomposition and parallelism) for each processor we use. Both criteria indicate that the parallel mesh generation method we present here is scalable and that we can generate billions of elements with insignificant overheads (see Table 5.3).

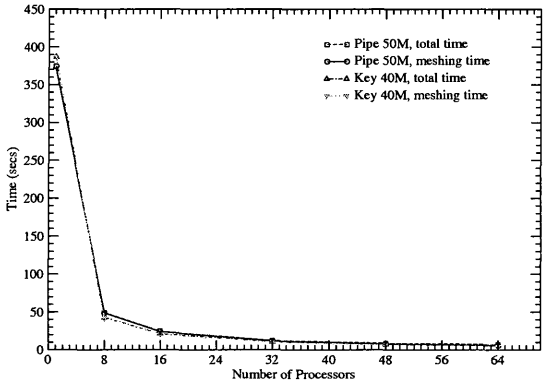


Figure 5.6: The performance for fixed size mesh.

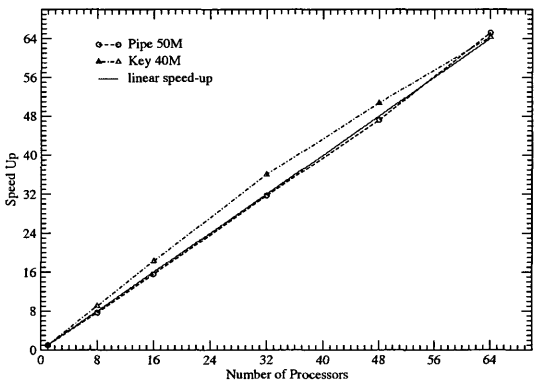


Figure 5.7: The speedup for fixed size mesh.

5.4.1 Fixed Size Mesh Experiments

In the fixed size set of parallel experiments we used a mesh of 40M elements for the Key domain and 50M for the cross section of the Pipe. For the key domain we created 12 subdomains for each processor while for the pipe 16 subdomains. The maximum triangle area is fixed throughout the experiments for each domain.

No of processors	1	8	16	32	48	64
The Domain						
No of subdomains	12	96	192	384	576	768
Mesh size (M)	43.32	43.34	43.37	43.41	43.43	43.45
PMADD time	0.20	0.37	0.44	0.60	0.83	1.05
Meshing time	386.32	42.35	20.72	10.12	6.79	4.96
Total time	386.52	42.72	21.16	10.72	7.62	6.01
The Pipe Domain						
No of subdomains	16	128	256	512	768	1024
Mesh size (M)	50.93	50.97	51.00	51.05	51.08	51.11
PMADD time	0.27	0.51	0.60	0.89	1.07	1.47
Meshing time	374.15	48.80	24.03	11.80	7.93	5.74
Total time	374.42	49.29	24.63	12.69	9.00	7.21

Table 5.2: Performance data for the key and the pipe geometry for a fixed maximum element area. All times are in seconds and mesh sizes are in millions (M).

The results are presented in Table 5.2. The data again indicate an unimportant increase in the number of elements for the different levels of over-decomposition, which shows that the over-refinement we introduce is insignificant. The total execution time and the computation time for the actual mesh generation are depicted in Figure 5.6. These times are very close, because the PMADD overhead cost is very small. This cost is neutralized by the effect of over-decomposition, which along with the good load balancing and zero communication during the parallel meshing, lead to superlinear speedup, see Figure 5.7. The speedup is calculated against the total time it takes to create the mesh on one processor, as it is presented in Table 5.2.

5.4.2 Scaled Size Mesh Experiments

A more practical way to evaluate the scalability and true performance of a parallel algorithm and software is to scale the size of the problem in proportion to the number of processors used. In the following experimental data we use the same level of decomposition for every configuration of processors, i.e., we keep the average number of subdomains per processor constant, and thus we eliminate the effect of over-decomposition in the resulting performance data. Theoretically we should be able to achieve the same creation time per element per processor for all the parallel configurations independently of the number of processors used. However, this is not feasible for the following two reasons: (1) the decomposition overhead, which increases very slowly but nevertheless there is an increase in the overhead as the number of processors increases and (2) load imbalances due to unpredictable and variable computation of the mesh generation kernel.

Table 5.3 shows some performance indicators for the two model problems we use, the key and the pipe geometry. In the experiments for the key model we created 12 subdomainns per processor and generated on average 1.6M elements per subdomain i.e., total 20M per processor. For the pipe model we created 20 subdomainns per processor and generated on average 1.6M elements per subdomain i.e., total 32M per processor. Small differences exist in the size of the mesh because our stopping criteria are based on the quality and size of elements, and thus the mesh size cannot be exactly predefined. It is clear from the Table 5.3 that for larger processor configurations, like

64 processors, the 99.5% of the total execution time is spent in the meshing phase by the Triangle. This suggests that for realistic problems the PMADD overhead is about 0.5% of the total execution time.

No of processors	1	8	16	32	48	64
The Key Domain						
No of subdomains	12	96	192	384	576	768
Mesh Size	20M	160M	320M	650M	860M	1.3B
Total time	152.43	177.31	192.41	213.91	166.10	205.26
Max meshing Time	152.23	176.92	191.93	213.26	165.25	204.19
Aver. meshing Time	152.23	165.75	168.04	170.31	137.70	163.14
Imbalance	1	1.067	1.142	1.252	1.200	1.252
MADD Phase time	0.20	0.38	0.44	0.63	0.84	1.05
Max MADD time	0.20	0.14	0.13	0.13	0.12	0.13
Tot. time/(elem./procs)	7.33	8.73	9.47	10.54	9.20	10.11
Additional Cost /procs	0%	2.4%	1.8%	1.4%	0.5%	0.6%
The Pipe Domain						
No of subdomains	20	160	320	640	960	1280
Mesh size	32M	240M	500M	1B	1.4B	2.1B
Total time	236.00	247.10	245.32	279.59	246.59	294.39
Max meshing time	235.71	246.53	244.65	278.56	245.09	292.71
Aver. meshing time	235.71	226.78	231.15	253.59	218.56	255.87
Imbalance	1	1.087	1.058	1.098	1.121	1.144
MADD phase time	0.29	0.55	0.67	1.01	1.48	1.66
Max MADD time	0.29	0.19	0.17	0.17	0.16	0.18
Tot. time/(elem./procs)	7.30	8.23	7.94	8.51	8.45	8.96
Additional Cost /procs	0%	1.6%	0.6%	0.5%	0.3%	0.4%

Table 5.3: Performance data for the key and the pipe geometry. The meshing time includes the time of the decoupling procedure (MADD). The MADD phase includes the load balance estimation procedure and the distribution of the subdomains to the processors. The imbalance is measured as ratio of the max meshing processor time over the average. All times are in seconds except for the time/(elem./procs) which is in microsecs.

We observe that, while the max PMADD time on one processor remains almost constant, the time for PMADD phase increases as the number of processors increases.

This is in agreement with the analysis in Section 4.1. As the number of processors increases, the number of PMADD iterations increases, although the number of the subdomains per processor is constant. In each PMADD iteration all the processors finish the decomposition, before the next iteration begins. This synchronization imposes an additional cost in the PMADD time. Moreover, the communication during this phase increases, as the number of processors increases. Fortunately, the communication and synchronization cost is less than 0.02 secs per processor. In comparison with the total execution time this cost is very small.

The load imbalance is measured by the ratio of the maximum meshing time on one processor and the average meshing time for all the processors. In Table 5.3 we observe that the load balance for the pipe is very good, 1.14 for 64 procs, while for the key is satisfactory, 1.25. The load-balance is based on over-decomposing the domain and equi-distributing the areas, and although it depends on the size of the mesh as we saw in Section 4.2, it also depends on the geometry and the number of the processors. Further improvement in the load-balance can be achieved by using parallel runtime software systems that address load-balancing problems, as the one presented in [2].

An important measure for evaluating the efficiency of a parallel meshing method is the (total) time spent for creating one element on one processor. Let $T^{(P)}$ be the total time running on P processors in order to create a mesh of size $S^{(P)}$. Then, the time per element, per processor is $T_e^{(P)} = \frac{T^{(P)} \cdot P}{S^{(P)}}$. This measure eliminates the differences in the mesh size, providing a more objective view of the scaled performance. We

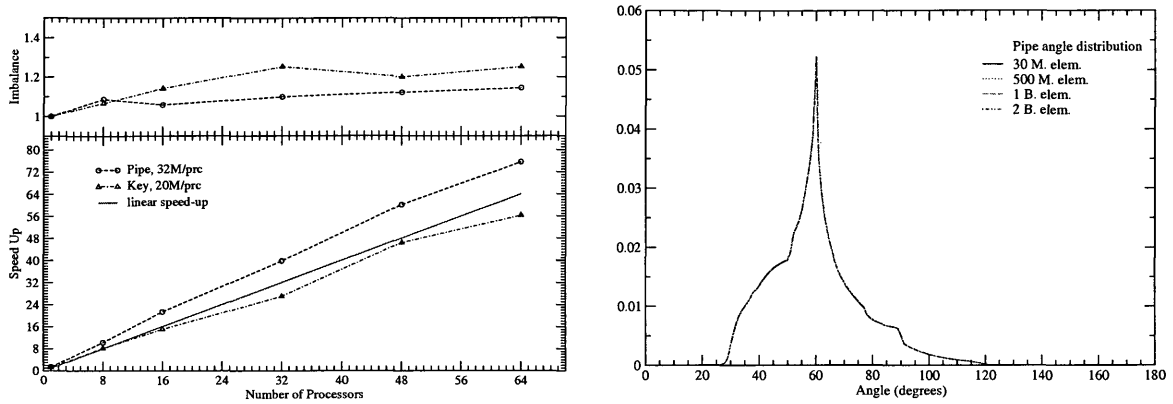


Figure 5.8: **Left: Top** is presented the imbalance and **down** the speedup for the scaled experiments. The speedup is measured against the sequential creation of 5M elements and is based on the overall time it takes for one element to be created. Observe the direct impact of the imbalance to the speedup. **Right:** The angle distribution for scaled mesh sizes of the pipe.

see in Table 5.3 that this time is almost constant, and thus the method is scalable. The slight increase of this time is mainly due to the imbalance increase, while the contribution of the overhead time cost is very small. This is evident in Figure 5.8, where the imbalance is depicted on the top and the scaled speedup down. The scaled speedup for P processors is measured as $U_P = \frac{T_e^s \cdot P}{T_e^{(P)}}$, where T_e^s is the time to create sequentially one element for a non-decomposed mesh of size 5M. We again observe the superlinear speedup for the same reasons as in the fixed size experiments. It is obvious in this figure the direct impact of the imbalance to the speedup.

Another measure for evaluating the scalability is the additional cost time cost for each processor that we use, relatively to the total time when running on one processor. The additional cost C_P per processor, when using P processors, is computed as $C_P = \frac{T_e^{(P)} - T_e^{(1)}}{T_e^{(1)} \cdot P}$. Taking into account that the mesh size $S^{(P)}$ is approximately proportional to the number of processors P, we have $C_P \simeq \frac{T^{(P)} - T^{(1)}}{T^{(1)} \cdot P}$. We can consider the quantity

$T^{(1)}$ as the ideal time for creating on P processors a mesh of size $S^{(P)} \simeq P \cdot S^{(1)}$, since the effect of over-decomposition is eliminated. In this way the additional cost C_P measures the distance from the ideal speedup, distributed to the number of processors used.

The time $T_e^{(P)}$ is increasing as P increases, the reasons were explained above. This increase though is small for the key and even smaller for the pipe domain. It is interesting to observe that the additional cost C_P tends to decrease, as P increases. Although we have to pay a (small) cost in the performance for each additional processor we use, this cost tends to decrease, when measured in scale. This result underlines the scalability of the method.

Finally we should compare the quality of the elements of scaled meshes that the decoupling procedure produces. In Figure 5.8 right is depicted the distribution of the angles of the elements, for meshes varying from 30M triangles to 2.1B. The quality is obviously the same.

Chapter 6

Conclusions and Future Work

We presented a decoupling procedure for parallel Delaunay guaranteed quality mesh generation on distributed memory machines for 2-dimensional domains. The method eliminates the communication during the mesh generation and demonstrates good stability in terms of the size and the quality of the final mesh. It also shows good speedup and scalability, making it suitable for creating very large meshes on distributed memory machines. A major advantage of the our method is that a sequential mesher (Triangle [44]) is used as a library, without any modification, achieving 100% code re-use. The method can be used at the same time as sequential mesh generation, in order to create larger meshes in less time using one processor. Because of the zero communication and the scalability for large meshes, this method seems to be suitable for Grid computing applications [16].

Future work for 2-dimensional geometries includes the theoretical analysis about over-refinement by using the local lfs (i.e., an adaptive way) to determine the local refinement of the decoupling zone. It is also interesting to see how this approach can

be applied in three dimensions for surface and volume parallel guaranteed quality mesh generation. The main issue in 3-dimensional domains is the creation of suitable domain decompositions, similar to the one we are able to create for the two dimensions.

Bibliography

[1] 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.

[2] K. Barker, N. Chrisochoides, A. Chernikov, and K. Pingali. A framework for load balancing of adaptive and asynchronous applications. *IEEE Trans. Parallel and Distributed Systems*, 14(12), 2003.

[3] K. Barker, N. Chrisochoides, J. Dobbelaere, D. Nave, and K. Pingali. Data movement and control substrate for parallel adaptive applications. *Concurrency and Computation Practice and Experience*, 14:77–101, 2002.

[4] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6:101–107, 1994.

[5] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, K. Moore, and V. Sunderam. PVM and HeNCE: Tools for heterogeneous network computing. In *Software for Parallel Computation*, J. S. Kowalik and L. Grandinetti, editors, volume 106. Springer-Verlag, 1993.

[6] G. E. Blelloch, G. L. Miller, J. C. Hardwick, and D. Talmor. Design and implementation of a practical parallel Delaunay algorithm. *Algorithmica*, 24(3/4):243–269, 1999.

[7] H. Blum. A transformation for extracting new descriptors of shape. In *Models for the Perception of speech and Visual Form*, pages 362–380. MIT Press, 1967.

[8] J. W. Brandt. Convergence and continuity criteria for discrete approximations of the continuous planar skeleton. *CVGIP:Image Understanding*, 59:116–124, 1994.

[9] J. W. Brandt and V. R. Algazi. Continuous skeleton computation by Voronoi diagram. *Comput. Vision, Graphics, Image Process.*, 55:329–338, 1992.

[10] L. P. CHEW. Constrained Delaunay triangulations. *Algorithmica*, 4:97–108, 1989.

[11] L. P. CHEW. Guaranteed quality triangular meshes. Technical Report TR-89-983, Department of Computer Science, Cornell University, 1989.

[12] L. P. CHEW. Guaranteed-quality mesh generation for curved surfaces. In *9th Annual Symposium on Computational Geometry*, pages 274–280. ACM, 1993.

[13] L. P. CHEW, N. CHRISOCHOIDES, AND F. SUKUP. Parallel constrained Delaunay triangulation. In *ASME/ASCE/SES Special Symposium on Trends in Unstructured Mesh Generation*, pages 89–96, 1997.

[14] H. CHOI, S. CHOI, AND H. MOON. Mathematical theory of medial axis transform. *Pacific Journal of Mathematics*, 181:57–88, 1997.

[15] N. CHRISOCHOIDES. An alternative to data mapping for parallel PDE solvers: parallel grid generation. In *Scalable Parallel Libraries Conference*, 1993.

[16] N. CHRISOCHOIDES, A. FEDOROV, B. B. LOWEKAMP, M. ZANGRILLI, AND C. LEE. A case study of optimistic computing on the Grid: Parallel mesh generation. In *Next Generation Systems Program Workshop, IPDPS'03*, 2003.

[17] N. CHRISOCHOIDES AND D. NAVE. Simultaneous mesh generation and partitioning. *Mathematics and Computers in Simulation*, 54(4-5):321–339, 2000.

[18] N. CHRISOCHOIDES AND D. NAVE. Parallel Delaunay mesh generation kernel. *International Journal for Numerical Methods in Engineering*, 58(2):161–176, 2003.

[19] N. CHRISOCHOIDES AND F. SUKUP. Task parallel implementation of the BOWYER-WATSON algorithm. In *Proceedings of Fifth International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields*, 1996.

[20] H. DE COUGNY AND M. SHEPHARD. Parallel volume meshing using face removals and hierarchical repartitioning. *Comp. Meth. Appl. Mech. Engng.*, 1999.

[21] B. N. DELAUNAY. Sur la Sphére Vide. *Izvestia Akademia Nauk SSSR, VII Seria, Otdelenie Matematicheski i Estestvennyka Nauk*, 7:793–800, 1934.

[22] R. DIEKMANN, D. MEYER, AND B. MONIEN. Parallel decomposition of unstructured FEM-meshes. *Concurrency: Practice and Experience*, 10(1):53–72, 1998.

[23] P. J. FREY AND P. L. GEORGE. *Mesh Generation.* Hermes Science Publishing, 2000.

[24] J. GALTIER AND P. L. GEORGE. Prepartitioning as a way to mesh subdomains in parallel. In *5th International Meshing Roundtable*, pages 107–122, 1996.

[25] P. L. GEORGE AND H. BOROUCHAKI. *Delaunay Triangulation and Meshing: Applications to Finite Element.* Hermis, Paris, 1998.

[26] H. N. GURSOY AND N. M. PATRIKALAKIS. An automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part i algorithms. *Engineering With Computers*, 8:121–137, 1992.

[27] B. HENDRICKSON AND R. LELAND. An improved spectral graph partitioning algorithm for mapping parallel computations. *SIAM Journal on Scientific Computing*, 16(2):452–469, 1995.

[28] B. HENDRICKSON AND R. W. LELAND. A multi-level algorithm for partitioning graphs. In *Supercomputing*, 1995.

[29] C. KADOW AND N. WALKINGTON. Design of a projection-based parallel Delaunay mesh generation and refinement algorithm. In *4th Symposium on Trends in Unstructured Mesh Generation*, 2003.

[30] G. KARYPIS AND V. KUMAR. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report TR 95-035, 1995.

[31] G. KARYPIS AND V. KUMAR. *MeTis: Unstractured Graph Partitioning and Sparse Matrix Ordering System, Version 2.0*, 1995.

[32] B. W. KERNIGHAN AND S. LIN. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49(2):291–307, Feb. 1970.

[33] C. L. LAWSON. Software for C^1 surface interpolation. In *Mathematical Software III*, John R. Rice, editor, pages 161–194. Academic Press, New York, 1977.

[34] D. T. LEE. Medial axis transformation of a planar shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI 4(4):363–369, July 1982.

[35] R. LOHNER AND J. CEBRAL. Parallel advancing front grid generation. In *International Meshing Roundtable.* Sandia National Labs, 1999.

[36] B. MAERTEN, D. ROOSE, A. BASERMANN, J. FINGBERG, AND G. LONSDALE. DRAMA: A library for parallel dynamic load balancing of finite element applications. In *European Conference on Parallel Processing*, pages 313–316, 1999.

[37] D. NAVE, N. CHRISOCHOIDES, AND L. P. CHEW. Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains. In *Proceedings of the 18th ACM Symposium on Computational Geometry*, pages 135–144. ACM Press, 2002.

[38] D. NAVE, N. CHRISOCHOIDES, AND L. P. CHEW. Guaranteed–quality parallel Delaunay refinement for restricted polyhedral domains. *Computational Geometry: Theory and Applications*, 2003.

[39] P. P. PEBAY AND P. J. FREY. A-priori Delaunay-conformity. In *7th International Meshing Roundtable*, pages 321–333, 1998.

[40] M. PRICE, C. STOPS, AND G. BUTLIN. A medial object toolkit for meshing and other applications. In *Proceedings of 4th International Meshing Roundtable*, pages 219–229, 1995.

[41] J. RUPPERT. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[42] R. SAID, N. WEATHERILL, K. MORGAN, AND N. VERHOEVEN. Distributed parallel Delaunay mesh generation. *Comp. Methods Appl. Mech. Engrg.*, 177:109–125, 1999.

[43] E. C. SHERBROOKE, N. M. PATRIKALAKIS, AND F.-E. WOLTER. Differential and topological properties of medial axis transforms. *Graphical Models and Image Processing*, 55(1):574–592, 1996.

[44] J. R. SHEWCHUK. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, Ming C. Lin and Dinesh Manocha, editors, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[45] J. R. SHEWCHUK. *Delaunay Refinement Mesh Generation.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1997. Available as Technical Report CMU-CS-97-137.

[46] T. K. H. TAM AND C.G ARMSTRONG. 2D finite element mesh generation by medial axis subdivision. *Advances in Engineering Software*, 56(13):313–324, 1991.

[47] C. WALSHAW AND M. CROSS. Mesh partitioning: A multilevel balancing and refinement algorithm. *SIAM Journal on Scientific Computing*, 22(1):63–80, 2000.

[48] C. WALSHAW, M. CROSS, AND M. G. EVERETT. Parallel dynamic graph partitioning for adaptive unstructured meshes. *Journal of Parallel and Distributed Computing*, 47(2):102–108, 1997.

[49] F.-E. WOLTER. Cut locus and medial axis in global shape interrogation and represenation. Technical report, MIT, Department of Ocean Engeneering, Design Laboratory, 1993.

VITA

Leonidas Linardakis

Leonidas Linardakis was born in Athens, Greece on November 1965. He received his B.S. at the University of Ioannina, Greece, in 1997 with a degree in Mathematics. He received his M.S. at the University of Ioannina in Mathematics in 1999.

In June 2001 the author entered the College of William and Mary as a graduate assistant in the Department of Computer Science. Leonidas Linardakis is currently a Ph.D. candidate and a Teaching Fellow in the Computer Science Department in the College of William and Mary.