OPTIMAL TWO-DIMENSIONAL TRIANGULATIONS

BY

TIOW-SENG TAN

B.Sc., National University of Singapore, 1984
M.Sc., National University of Singapore, 1988

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1993

Urbana, Illinois

Optimal Two-Dimensional Triangulations

Tiow-Seng Tan, Ph.D.
Department of Computer Science
University of Illinois at Urbana-Champaign, 1993
Professor Herbert Edelsbrunner, Advisor

A (*geometric*) *triangulation* in the plane is a maximal connected plane graph with straight edges. It is thus a plane graph whose bounded faces are triangles. For a fixed set of vertices, there are, in general, exponentially many ways to form a triangulation. Various criteria related to the geometry of triangles are used to define what one could mean by a triangulation that is optimal over all possibilities. The general problem studied in this thesis is the following:

> given a finite set S of vertices, possibly with some prescribed edges, how can we choose the rest of the edges to obtain an optimal triangulation?

Just to mention an example, we are interested in computing a *min-max angle triangulation* of S, that is, a triangulation whose maximum angle over all its triangles is the smallest among all triangulations of S.

This thesis presents a number of new algorithms to construct optimal triangulations useful in engineering and scientific computations, such as finite element analysis and surface interpolation. All algorithms are the first and, currently, the only ones that construct the defined optimal triangulations in time polynomial in the input size. These main results are described in three parts.

First, we develop a new algorithmic technique called the *edge-insertion paradigm*. It computes for a set of n vertices an optimal triangulation defined by some generic criterion. From this, we deduce that a min-max angle and a *max-min height triangulation* can be computed in $O(n^2 \log n)$ time and linear storage, and a *min-max slope* and a *min-max eccentricity triangulation* in cubic time and quadratic storage.

Second, we show that a *min-max length triangulation* for a set of n vertices can be computed in quadratic time and storage. Length refers to edge length and is measured by some normed metric such as the Euclidean or any other l_p metric.

Third, for a given plane graph of n vertices and m non-crossing edges, we prove that there is a set of $O(m^2 n)$ points so that, for each adjacent pair of points on an edge, there exists a circle passing through the two points that encloses no other points. This implies an efficient way to construct a so-called *conforming Delaunay triangulation*, which is a Delaunay triangulation that subdivides the given plane graph.

These results collectively provide a foundation for further algorithmic studies of optimal triangulations.

To
my parents Tan Bac Hik and Lee Gee Moi,
and
my wife Chia Joo Joo
for their love.

Acknowledgments

Thank you very much—teachers, friends, and family. You in various ways have helped shape this work from beginning to end. It has been one very long, challenging and arduous adventure, and I sincerely appreciate all the help and advice that you have kindly bestowed upon me along the way.

To the National University of Singapore, I give thanks for the study leave and the financial support under the Oversea Graduate Scholarship. I certainly have a fruitful and memorable five-year stay at the University of Illinois at Urbana-Champaign.

Professor C. L. Liu, I thank you for instilling in me the dating principle in doing research. Your support, guidance and encouragement during the first two years have built up my perseverance in venturing date after date in the subsequent years. Indeed, no results might have been obtained, if not for the experiences learned from failures. My VLS thanks to you, Dave.

Professor Douglas West, I thank you for many lessons on proofs. In particular, your first fifty minutes lecture on proof techniques (Math 312) in August 1988 was most enlightening. Your lessons have served as the fundamental preparation in my first theoretical attempt in Spring 1989. Thank you, Doug; I enjoyed your teaching all these years (until I was carried away by other graphical drawings).

Professor Herbert Edelsbrunner, thesis advisor and friend, you have been the inspiration to the work in the past four years. Your visions in the area set the beginning of this thesis, and your involvement in working through vague ideas produced building blocks to the work. The journey with you in discrete and computational geometry, and at Urbana, Champaign, Graz, Vienna and Berlin, has been wonderful, exciting, and memorable. Herbert, my multi-dimensional thanks to you!

To the members of the thesis committee, Professor Herbert Edelsbrunner, Professor C. L. Liu, Professor Faisal Saied, Professor Pravin Vaidya, and Professor Douglas West, I thank you all for your service and interests. And, thank you, Herbert Edelsbrunner, Edgar Ramos, and Douglas West, for many suggestions and corrections to parts of this thesis, and, Marshall Bern, Herbert Edelsbrunner, David Eppstein, Scott Mitchell, and Roman Waupotitsch, for our joint work.

To past and present members of our research group, Nataraj Akkiraju, Ulrike Axen, Cecil Delfinado, Tamal Dey, Herbert Edelsbrunner, Mike Facello, Klaus Jansen, Sang Ho Lee, Pat Moran, Ernst Mücke, Edgar Ramos, Nimish Shah, and Roman Waupotitsch, I thank you all for useful meetings, and especially to you, Roman, for sacrificing many fine afternoons to breath chalk dust at DCL 2429. And, Joyce Woodworth, thank you for your help in administrative matters.

It was a stressful two months in preparing for the qualifying examination (December 1988–February 1989). To my friends, Ran Libeskind-Hadas, for your inspiration and company in the preparation, and Nany Hasan, for your encouragement and support, and to my wife Joo-Joo, for your forgiveness and quiet acceptance to the bully at home, I thank you all so very much!

To many friends, I thank you for your letters and sweet memories together at Urbana-Champaign. Thank you, Chuah Ban Keat, Lim Pink, Low Chin Kok, Tan Mong Kia, and Toh Sew Kiok, for your many traditional communications. Thank you, Chua Chong Seng and family, Khoo Siau Cheng, Lau Thiam Beng, Uncle Lee Ching Chic, Leong Hon Wai, Lim Kok Yau and family, Low Chin Kok, Whee Keng and Tan Kok Phuang, and Yuen Chung Kwong, for your long distance visits. Thanks you, Poh Lin and Kee Ching Leng, Lee Chia Ling, Lay Chin and Lee Hing Yan, Jessica and Leong Hon Wai, Mui Joo and Ng Pin, and Clover and Kelvin Sung Hsien Ching, for tips on home abroad.

And, thank you, Chong Chow Pin, Jocelyn Chong, Chua Teck Joo, Olga Fishman, Foo Soo Mee, Fu Ping, Donald Gillies, Hung Gih Guang, Teck Choo and Stephen Kwek, Dennis Lee, Yen Kim and Lee Meng Chua, Albert Li, Lim Hock Beng, Lim Siew Siew, Esther and Stephen Lim, Lim Swee Boon, Lim Yeow Beng, Loh Mei Yun, Look Jee Loon, Sharon and Low Chin Chau, Mok Chee Liang, Ong Leong Seng, Shi Weiping, Sim Hock Kee, Tan Jin Ho, Carol and Tan Kok Wah, May Ling and Tia Too Seng, and Hsuehling and Yang Der Shung, for wonderful times together.

To my family and in-laws, your moral supports have been overwhelming. Thank you all very much! Special thanks to you, Joo Joo, for your love.

Tiow Seng

Thanksgiving Day, 1992

TABLE OF CONTENTS

Chapter

Chapter 1

Introduction

The theme of this thesis is geometric triangulations. These are face-to-face decompositions of domains or spaces into triangles in two dimensions or simplices in general dimensions. Computational work on geometric triangulations, or simply, triangulations, has been plentiful, ever since the beginning of the computer era. Many interesting theoretical properties have been discovered about triangulations, which in turn suggest the use of particular types of good or optimal ones. Unfortunately, many of these optimal triangulations have as yet no polynomial time construction methods and thus remain of limited significance in practice. Many computer programs written in the area are either for optimal triangulations of a small class or heuristics with no guarantee on the quality of the output triangulation. There remain numerous unsolved problems on constructing triangulations, all waiting anxiously for solutions. The pool of problems that was open a few years ago motivated our study, and this thesis reports on some solutions and encouraging results in two dimensions.

In the rest of this chapter, we discuss two representative applications of triangulations (Section 1.1), introduce useful terminology (Section 1.2), survey open problems related to applications (Section 1.3), and preview material covered by subsequent chapters (Section 1.4).

1.1 Two Sample Application Areas

Triangulating a geometric object or space is a popular decomposition method that has found many applications. In this section, we describe two major uses of triangulations; other applications will be mentioned where appropriate in later discussions. The main purpose here is to shed some light on the role of triangulations that will help us understand the relevance of the open problems discussed in Section 1.3. As such, we only highlight the typical steps involved in applications; specific details and other variations can be found in the references[1].

FINITE ELEMENT ANALYSIS. This is a discretization technique for solving partial differential equations. Many problems in engineering, including elasticity, electromagnetism, fluid dynamics, structural mechanics, biomechanics and many others, appear as variational forms where in each case a function is to be found that minimizes or maximizes a given set of equations. Very often, an exact solution is impossible and thus an approximation is necessary. Finite element analysis is one such approximation method [BaSu90, StFi73].

The analysis starts by creating a so-called *mesh*—possibly a triangulation—that decomposes the structure or the object of interest into small pieces called *elements*. Next, the partial differential equation is solved for each element with some lower order polynomial function. The solutions for the elements are then integrated into a solution for the problem. Then, some error indicator is obtained to decide whether any further computation is required to improve the current solution. The next iteration, if needed, will be performed with higher order polynomial functions or with improved mesh topology (via relocating the mesh vertices or refining elements into smaller ones), or combination of both.

The mesh generation is indeed an important process. The overall quality of a computation depends on the types and shapes of the elements, and the number and distribution of the elements. One way to generate a mesh is to first spread vertices on the object, possibly taking distribution requirements into account, and then form non-overlapping elements using these vertices [Cave74]. Another way is to generate elements one at a time from the remaining object [GeSh90, Trac77]. Sometimes, the object is first decomposed into some manageable, for

[1] Because of the huge amount of literature in the area, the references quoted are by no means complete.

example, convex portions [Joe86, JoSi86] before a mesh is created. We refer to [HoLe88, Shep88] for surveys on mesh generation methods.

Surface Interpolation. The objective is to construct a surface that passes through a given set of data points with corresponding data values. This process appears in a wide variety of scientific fields including mineral exploration, computer aided geometric design, medicine, digital terrain modeling and weather analysis. In these applications, the given points and values represent observed or computed values of some physical phenomena, and the interpolation schemes are used to construct surfaces that approximate other values, which may be difficult or impossible to obtain. Conditions, such as the continuity of partial derivatives (i.e., smoothness) of the surface, are usually assumed to limit the choice. A comprehensive list of references on the subject can be found in [FrSc87].

Let us consider the example of a given set of planar data points (x_i, y_i) with corresponding data values z_i. The goal is to construct a bivariate function $f(x, y)$ such that $f(x_i, y_i) = z_i$, for all i. A typical three-step interpolation procedure to achieve this is as follows. First, construct a triangulation on the set of data points to divide the region of interest into triangles. Second, estimate partial derivatives of f with respect to x and y at each of the data points (x_i, y_i) using the data values of the data points connected to (x_i, y_i) in the triangulation. The plausible justification for taking the neighbors of (x_i, y_i) is that they tend to be close to (x_i, y_i) and should thus have an influence on this calculation. Third, interpolate the values of points within each triangle by some polynomial using the data values and the estimated partial derivatives at the vertices of the triangle. The surface f is then obtained by connecting the interpolated results on all the triangles, and some error bounds on the goodness of f can also be computed from the geometry of the triangles.

1.2 Terminology

This section summarizes basic definitions and concepts needed throughout this thesis. The discussion is predominantly for the two-dimensional plane, $\mathbb{R}^2$, because results and problems of interest here are mostly two-dimensional.

Convexity. Denote by xy the edge, or line segment, that connects the points $x, y \in \mathbb{R}^2$, and by $|xy|$ the *length* of xy. The *endpoints* of xy are points x and y; they are, however, not considered as points on xy. For $x, y, z \in \mathbb{R}^2$, xyz denotes the triangle with edges xy, yz and zx. A set $C \subseteq \mathbb{R}^2$ is *convex* if for $a, b \in C$, ab lies entirely in C; otherwise, C is *non-convex*. The convex hull of a set S, denoted by $conv(S)$, is the smallest convex set in $\mathbb{R}^2$ that contains S. A *region* refers to the set of points in $\mathbb{R}^2$ bounded by a (simple) closed curve. The curve is also the *boundary* of the region. For example, a *disk* is a region bounded by a circle. We always refer to a region as a *open* set that does not contain points of its boundary, and use the *closure of a region* to mean the region plus its boundary. A *convex region* is a region whose set of points is convex.

Plane Geometric Graphs. Let S be a finite set of n vertices, or points, in $\mathbb{R}^2$, and E a set of edges determined by S. We call $\mathcal{G} = (S, E)$ a *plane geometric graph* if

(i) for every edge $ab \in E$, $ab \cap S = \emptyset$, and

(ii) for every two edges $ab \neq cd$ in E, $ab \cap cd = \emptyset$.

If $E = \emptyset$, we refer to $\mathcal{G}$ simply as a (finite) *vertex set*, or *point set*, S. The connected components of $\mathbb{R}^2$ minus all vertices of S and all points on edges of E are the *faces* of $\mathcal{G}$.

Triangulations. A *triangulation* is a plane geometric graph $\mathcal{T} = (S, E)$ so that E is maximal. By maximality, edges in E bound $conv(S)$ and divide its interior into disjoint faces bounded by triangles. These triangles are referred as *triangles of* $\mathcal{T}$. We sometimes write $\mathcal{T}(S)$ to refer specifically to a triangulation $\mathcal{T}$ of S. As $\mathcal{T}$ is a connected plane graph, its $|S| = n$ vertices, $|E| = e$ edges, and f faces satisfy Euler's formula: $n - e + f = 2$; see, for example, [BoMu76]. In addition, if h is the number of edges in E bounding $conv(S)$, then $3(f-1) + h = 2e$, by counting the edges in two ways. The left hand side counts 3 for each bounded face and h for the unbounded face; the right hand side counts 2 for each edge. We thus obtain $e = 3n - h - 3$ and $f = 2n - h - 1$. Because h is fixed for a fixed point set S, it follows that all triangulations of S have the same number of $f - 1$ triangles and e edges.

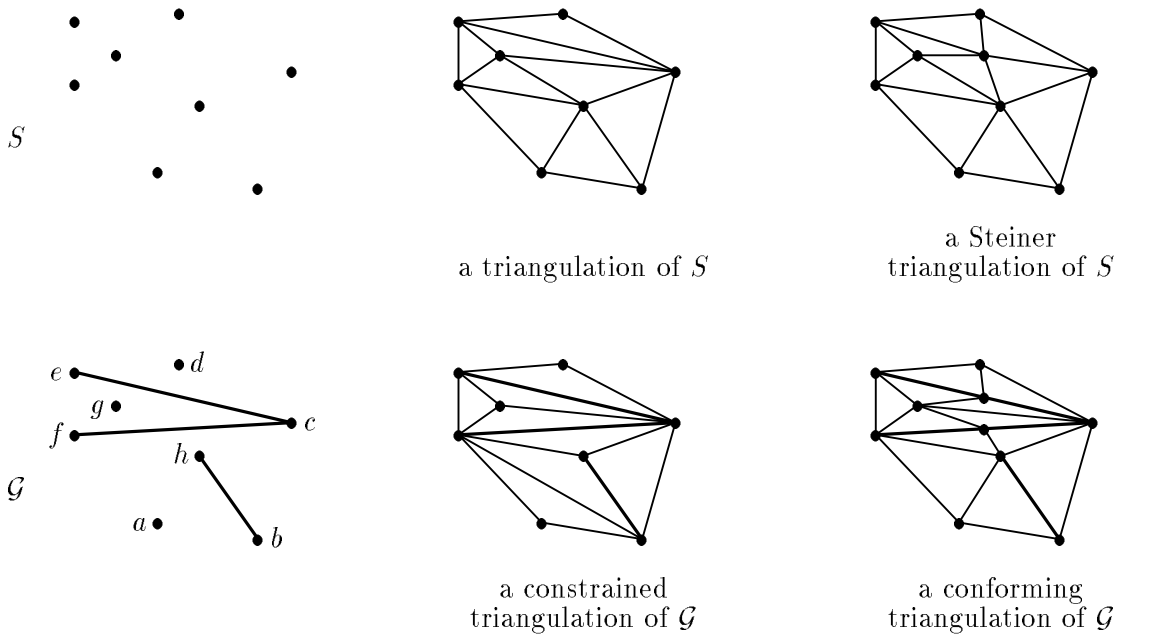

Figure 1.1: Examples on various types of triangulations for the point set S and the plane geometric graph $\mathcal{G} = (\{a, b, c, d, e, f, g, h\}, \{bh, ce, cf\})$.

A plane geometric graph $\mathcal{G} = (S, E)$ can be augmented with an edge set E' until it is a triangulation $\mathcal{T} = (S, E \cup E')$, also referred to as a *triangulation of* $\mathcal{G}$. If $E \neq \emptyset$, we call $\mathcal{T}$ a *constrained triangulation* (of $\mathcal{G}$), and E its set of *constraining edges*. Besides with edges, we can also augment $\mathcal{G} = (S, E)$ with a vertex set S'. A triangulation $\mathcal{T}$ obtained in this manner is called a *Steiner triangulation* (of $\mathcal{G}$), and S' the set of *Steiner vertices*. We call a Steiner triangulation with constraining edges a *conforming triangulation*. Refer to Figure 1.1 for an illustration. Unless stated otherwise, we talk about triangulations without Steiner vertices and without constraining edges.

Polygons. A (simple) *polygon* is a plane geometric graph $\mathcal{G} = (S, E)$ where E and S form a single cycle. This cycle is the *boundary* of the polygon. It divides the plane into a bounded face, its *interior*, and an unbounded face, its *exterior*. We use polygon interchangeably to denote both the boundary (a curve) and its interior (a region or, more precisely, a *polygonal region*). A polygon is *convex* if its interior is convex, and it is called an n-gon if $|S| = n$. For example, triangles are convex 3-gons. We define *quadrilaterals* to be 4-gons, and *pentagons* to be 5-gons. Vertices of a polygon are always written in the order they appear on the boundary. Two vertices are *adjacent* if they are incident to a common edge. A *diagonal* of a polygon is a line segment that connects two non-adjacent vertices and lies entirely in the interior. An *ear* of a polygon is a triangle bounded by two edges and one diagonal [Meis75]. Treating E as a constraining edge set, we can talk about constrained and Steiner triangulations for $\mathcal{G}$. The restriction of one of these triangulations to the interior face of $\mathcal{G}$ is a *polygon triangulation.*

OPTIMAL TRIANGULATIONS. A plane geometric graph $\mathcal{G}$ permits, in general, many possible *triangulations*. Various (shape) criteria can be used to classify some as *optimal triangulations.* Many of these criteria are defined as *max-min* (short for maximizes the minimum) or *min-max*[2] of some triangle or edge *measure*. The first quantifier is over all triangulations of $\mathcal{G}$ and the second is over all triangles or edges of a triangulation. Two examples of criteria are min-max angle and max-min angle. A criterion may or may not define a unique optimal triangulation. When Steiner vertices are allowed, we prefer optimal triangulations with small number of vertices, for obvious computational reasons.

A max-min or a min-max criterion, in the case without Steiner vertices, can be extended naturally to its vector form. For example, consider the min-max angle criterion. For a triangulation $\mathcal{A}$ of $\mathcal{G}$, we define its angle vector $V_{\mathcal{A}} = (\alpha_1, \alpha_2, \ldots, \alpha_t)$, with $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_t$ the t largest angles of the $t = f - 1$ triangles of $\mathcal{A}$. If $\mathcal{B}$ is another triangulation of $\mathcal{G}$ with angle vector $V_{\mathcal{B}} = (\beta_1, \beta_2, \ldots, \beta_t)$, we say $V_{\mathcal{B}}$ is (lexicographically) smaller than $V_{\mathcal{A}}$ if there is an index $1 \leq j \leq t$ so that $\beta_i = \alpha_i$ for $1 \leq i < j$ and $\beta_j < \alpha_j$. Then, the vector form of the criterion in this case defines a triangulation with the minimum angle vector. Similarly, the vector form of the max-min angle criterion defines a triangulation that lexicographically maximizes the non-decreasing vector of smallest angles of triangles.

Additional optimal triangulations will be introduced in the next section. For now, we define the most prominent optimal triangulation of a point set S, named after the Russian mathematician Boris Delaunay.

DELAUNAY TRIANGULATIONS. An edge ab, for $a, b \in S$, is a *Delaunay edge* if there is a circle through a and b so that all other points of S lie outside the circle. The collection of Delaunay

[2] Though min-min and max-max versions are possible, they are usually trivial and uninteresting.

edges defines a plane geometric graph $\mathcal{D}(S)$ known as the *Delaunay triangulation* of S [Dela34]. In the non-degenerate case, which excludes four or more points on a common circle, $\mathcal{D}(S)$ is indeed a triangulation. In fact, it is a triangulation that lexicographically maximizes the non-decreasing vector of smallest angles of triangles. In the degenerate cases, some faces of $\mathcal{D}(S)$ are convex polygons other than triangles, and these can further be subdivided into disjoint triangles using additional edges cd for $c, d \in S$. The resulting triangulation is called a *completion* of $\mathcal{D}(S)$. In many situations, we make no distinction between Delaunay triangulations and their completions. Note that each edge cd of a completion satisfies the so-called *empty disk property* (with respect to S), i.e., there is a circle through c and d so that all other points of S lie outside the (open) disk bounded by the circle.

For a general plane geometric graph $\mathcal{G} = (S, E)$ where $E \neq \emptyset$, we extend the above definition to define constrained and conforming Delaunay triangulations. Two vertices a and b are *visible* from each other if the line segment ab does not intersect any constraining edges in E. A *constrained Delaunay triangulation* of $\mathcal{G}$ is a triangulation $\mathcal{T} = (S, E \cup E')$, where $ab \in E'$ if a and b are visible from each other and ab satisfies the empty disk property with respect to only vertices visible from both a and b [LeLi86]. A *conforming Delaunay triangulation* of $\mathcal{G}$ is a completion of $\mathcal{D}(S \cup S')$ where each edge of $\mathcal{G}$ is the union of some edges and vertices of the completion [BFL88].

<u>Time- and Storage-Complexity</u>. The efficiency of an algorithm is measured by its time- and storage-complexity. *Time* refers to the number of steps, as a function of the input size n, needed to complete the computation. *Storage* refers to the amount of memory space needed and is also measured as a function of n. We assume the *random access machine* as our computational model and use the *unit-cost measure* which charges one unit of time per arithmetic operation [AHU74]. As a common practice, we talk about the asymptotic order or order of magnitude of a complexity function $f(n)$ rather than the exact function itself. Thus, we write

(i) $f(n) \in \mathrm{O}(g(n))$ if there is a positive constant c so that $f(n) \leq c \cdot g(n)$ for all integers n exceeding some constant $n_0 \geq 0$,

(ii) $f(n) \in \Omega(g(n))$ if $g(n) \in \mathrm{O}(f(n))$, and

(iii) $f(n) \in \Theta(g(n))$ if $f(n) \in \mathrm{O}(g(n))$ and $f(n) \in \Omega(g(n))$;

see, for example, [PrSh85, page 8–10].

The terms *linear, quadratic, cubic* refer to polynomials in n with degree 1, 2, and 3 respectively. A *polynomial time algorithm* is an algorithm whose time-complexity is some polynomial in the input size n. For practical purposes, efficient algorithms commonly refer to those with low order polynomial time such as linear, $\mathrm{O}(n \log n)$, and quadratic. A computational problem is said to be in the class P if it has a polynomial time algorithm, specifically, *deterministic polynomial time algorithm.* Besides P, the class NP is of interest here. A problem is in NP if it has a polynomial time, *non-deterministic* algorithm (see [GaJo79] for detailed discussion). P is in NP but the converse relation is not known. There are decision problems in NP known to be equivalent, by a polynomial time transformation, to all other problems in NP. Such problems are termed *NP-complete* problems, and are unlikely to be in P.

<u>DATA STRUCTURES</u>. Data structures are ways to organize data in a computer. Common examples of data structures are arrays, stacks, priority queues, and trees. These are well-known and discussed in many standard texts on algorithms; see, for example, [CLR90]. We next introduce the *quad-edge* data structure [GuSt85] to store a plane geometric graph $\mathcal{G}$. Loosely speaking, it stores each edge of $\mathcal{G}$ four times, twice for its two incident faces and twice for its two endpoints[3]. Figure 1.2 shows an example of the quad-edge structure for a plane geometric graph that bounds polygonal regions A and B. Strictly speaking, A is not a polygonal region in the usual sense of the term since A contains a "crack" due to edges fg and gh. The quad-edge representation treats A as a single region bounded by those (oriented) edges for which it lies to their left. As such, we need not distinguish a region with cracks from a genuine one as long as its interior remain *simply connected.* By this we mean that different (simple) paths (disjoint from the edges) between two points in the interior can be continuously deformed into each other.

[3] Often, we do not need all the four (oriented) versions of an edge, but rather only the two versions that correspond to the endpoints of the edge.

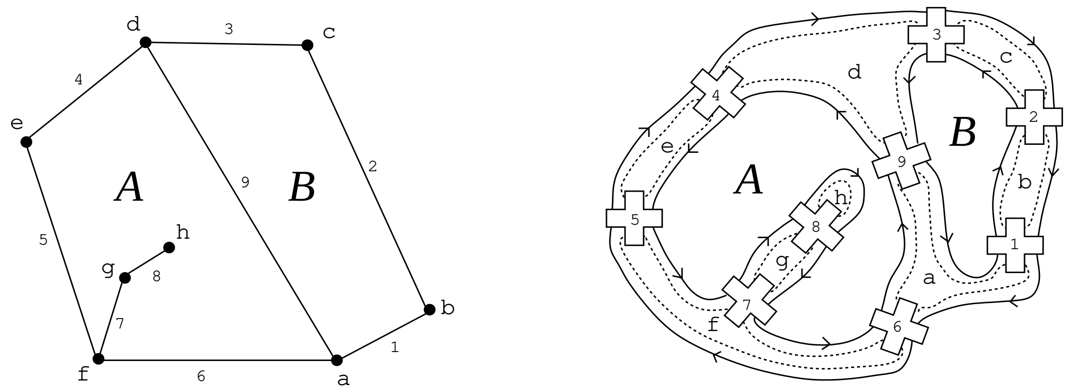

Figure 1.2: To the right is a schematic diagram of a quad-edge representation of the plane geometric graph to the left. Quad-edge records are shown represented by crosses which are linked by solid and dotted curves.

1.3 Survey of Problems

In the following subsections, we describe questions about constructing optimal triangulations. Some of these questions are answered in this thesis, and some remain open. Our emphasis is on efficiency, that is, algorithms that run in low order polynomial time. As a fixed set of n vertices generally has exponentially many triangulations (an upper bound of 10^{13n} on the number of triangulations of a set of n points in $\mathbb{R}^2$ is known [ACNS82]), it is not feasible to exhaustively search the set of all triangulations for an optimal one. Also, many straightforward ideas usually do not work as we will see in later chapters. The same can be said about Steiner triangulation problems. Most existing techniques for constructing optimal triangulations are surveyed in Chapter 2.

Sections 1.3.1 to 1.3.4 collect problems on non-Steiner triangulations, and Section 1.3.5 on Steiner triangulations. We note that some non-Steiner cases can be extended naturally to constrained or Steiner cases. Also, max-min or min-max criteria mentioned in these problems can be substituted by their corresponding vector forms. As mentioned, all problems are stated as questions on the existence of efficient algorithms, particularly, sequential ones. The same questions can also be asked in relation to other computational issues, such as parallel computation, dynamic inputs, etc. These questions will not be discussed.

1.3.1 Angle Criteria

For many applications, a "good" triangulation is one without thin or elongated triangles. Put it differently, triangles that differ as little as possible from equilateral triangles are usually preferred. One way to capture this notion is to impose angle criteria such as min-max angle or max-min angle. This indeed agrees with the fact that angle is an important quantity in bounding the errors of computations in finite element analysis and surface interpolation. In particular, there are bounds that relate the error directly to the size of smallest angles, and others that relate it to the size of the largest angle; see, for example, [Akim84, BaAz76]. Delaunay triangulations optimize the max-min angle criterion, and can be computed in $\Theta(n \log n)$ time by a number of algorithms (Section 2.4). On the other hand, the construction problem for min-max angle criterion was open for some time [Hans90]. An efficient solution to the problem is provided in Section 4.1.

Problem 1 How fast can we compute a min-max angle triangulation of a point set?

Besides min-max and max-min angle conditions, a limit on the range of the angle values is another natural condition. We note that this, however, may not always be achievable without introducing Steiner vertices. For example, it is not always possible to use only acute angles. The next question is a variant of this general issue, and is still open at this day.

Problem 2 Given a point set S, how fast can we compute a triangulation of S that minimizes the number of obtuse angles?

1.3.2 Length Criteria

For finite element analysis, edge lengths are sometimes part of error bounds. Specifically, the sharpness of some error bounds is inversely proportional to the longest side of a triangle [BrZl70, WGS90]. For surface interpolation, we sometimes use vertices of a triangle as "nearby" locations to estimate the data values of points in its interior. It is thus desirable to have interior points of the triangle as close to its vertices as possible [Akim84, page 44]. Also, this may help in estimating partial derivatives of other vertices connected to the triangle. Both applications suggest the criterion of min-max length. Of course, this is ignoring the tradeoff and interplay between angle and length criteria; they may not be optimized simultaneously. We can still formulate the problem on min-max length criterion (which also appears in [PeKe87, page 175], [Schu87, page 221], and [WaPh84, page 218]), and provide a solution in Chapter 5.

Problem 3 How fast can we compute a min-max length triangulation of a point set?

The following problems on length criteria are still open. Problem 4 is the "reverse" of the previous problem, and Problem 5 is notoriously difficult [Lloy77, PlHo87] (see Section 2.5 for some general discussion).

Problem 4 How fast can we compute a max-min length triangulation of a point set?

Problem 5 Given a point set S, how fast can we compute a triangulation of S that minimizes the sum of edge lengths?

1.3.3 Other Reasonable Criteria

As mentioned, error bounds are sometimes expressed in terms of angles as well as edge lengths. We can thus consider to optimize measures that relate, in some ways, to both. These include, for instance, the area of a triangle and the aspect ratio. The latter is the ratio of the longest edge to the altitude from this edge. The degree of a vertex is yet another interesting measure as it signifies the importance of the vertex in computations [FrFi91]. It is, however, an NP-complete problem to decide whether a point set with constraining edges has a triangulation with vertex degree at most 7 [Jans92]. No solution is known for the next problem compiled from [Schu87, page 222], [Barn77, page 84], [GeSh90, page 202], and [GCR77, Lind83].

Problem 6 Can a min-max or a max-min optimal triangulation based on any one of the following quality measures be computed efficiently: area; aspect ratio; degree; radius of inscribed circle; ratio of the area of the inscribed circle to the area of the triangle; ratio of the diameter of the inscribed circle to the radius of the circumscribed circle?

Two other measures are intentionally left out from the above; these are the max-min height [GCR77] and the min-max eccentricity [GCR77, WaPh84]. *Height* of a triangle refers to the length of the altitude from its longest side, and *eccentricity* of a triangle refers to the infimum over all distances between the center of its circumscribed circle to points in the closure of the triangle. These are formulated separately as they now have efficient solutions discussed in Sections 4.2 and 4.4.

Problem 7 How fast can we compute a max-min height triangulation of a point set?

Problem 8 How fast can we compute a min-max eccentricity triangulation of a point set?

1.3.4 Data Dependent Criteria

Notice that all the above mentioned optimal triangulations for use in interpolating $z = f(x, y)$ do not take the data values into account. This may not be desirable since some shape information "encoded" in the data values has great influence on the quality of the interpolation. Indeed, a study shows that the best triangulation should have triangles with long edges in the direction of minimum curvature and short edges in the direction of maximum curvature, rather than simply having triangles close to equilateral ones [Nadl85]. This requirement is an example of a data dependent criterion. In general, a *data dependent criteria* is one that takes into account more than just the locations of the vertices.

Work done on data dependent criteria can be found in [DLR90b, QuSc90, Ripp92, RiSc90]. We sample only two such criteria for interpolating $z = f(x, y)$. They are the min-max AABNV, short for *acute angle between normal vectors* [DLR90a, DLR90b], and the min-max slope [WaPh84, page 218]. To understand these, we imagine that a triangulation on points (x_i, y_i) in the plane is actually a "spatial triangulation" on (x_i, y_i, z_i) in space, i.e., a terrain formed by triangular faces. The intrinsic dimension of this triangulation is two and it lives in $\mathbb{R}^3$. It should be clear that we can talk about *normal vectors* and *slope* of these faces with respect to the plane $z = 0$. This naturally defines the mentioned criteria. The plausible justification for these criteria is that they model gradual change in the z values, in particular z values across faces, and thus better control the smoothness condition across faces in the interpolation.

Problem 9 Can an optimal triangulation defined by any one of the following criteria be computed efficiently: min-max AABNV, minimum sum of AABNV, and minimum sum of the squares of AABNV?

Problem 10 How fast can we compute a min-max slope triangulation of a point set?

Problem 9 is still open; heuristic algorithms are used in practice [DLR90a]. On the other hand, there is an efficient algorithm for problem 10 in Section 4.3.

1.3.5 On Steiner Triangulations

Many open problems on Steiner triangulations are summarized in [BeEp92]; we will highlight only two here. We note that constraining edges are commonly used to model important features of objects or spaces to be triangulated, and they constitute part of the edges in the resulting triangulation. Popular optimal triangulations with constraining edges include the constrained and conforming Delaunay triangulation. There are already efficient construction algorithms for the former [Chew89, Seid88], but not for the latter [NaSr91].

Problem 11 How fast can we compute a conforming Delaunay triangulation for a plane geometric graph?

Chapter 6 contains a polynomial time solution to the problem. The next open problem is related [BeEp91], since a triangulation without obtuse triangles is indeed Delaunay.

Problem 12 How fast can we compute a conforming triangulation without obtuse triangles for a plane geometric graph?

1.4 Overview of the Thesis

The following chapters discuss algorithms on computing optimal triangulations. Chapter 2 surveys some prior work relevant to our subsequent discussions in Chapters 3 through 6.

Chapter 3 discusses a new algorithmic technique, called the *edge-insertion paradigm*, which computes optimal triangulations for plane geometric graphs. We present an abstract view of the paradigm and state two conditions for criteria that can be optimized. Four examples of criteria known to satisfy these conditions are the min-max angle, max-min height, min-max slope and min-max eccentricity; these are discussed in Chapter 4. The material of these two chapters also appears in [ETW90, BEEMT92].

Chapter 5 presents a quadratic time solution to the min-max length triangulation problem of a point set. The result is applicable to edge lengths measured by an arbitrary normed metric, including the Euclidean distance and the more general l_p metrics. It is currently the only (non-trivial) length criterion that can be computed efficiently. Our solution also provides additional insight into optimal triangulations under edge length criteria. This chapter also appears in [EdTa91].

Chapter 6 considers conforming Delaunay triangulations. We show that, for every plane geometric graph $\mathcal{G}$ with n vertices and m edges, there is a conforming Delaunay triangulation for $\mathcal{G}$ with $4m^2n + 10mn + 4n$ vertices. The result also implies an efficient algorithm to compute these vertices and thus a conforming Delaunay triangulation. This is also published in [EdTa92].

We note that all solutions found in the above chapters are reasonably simple and not exceedingly difficult to implement. They are currently the only solutions of their kinds that run in polynomial time. The data structures needed are mainly arrays, stacks, priority queues, trees, and quad-edges. For each algorithm the tricky part is its correctness, which is based on subtle observations and has a relatively involved and lengthy proof.

Chapter 7 summarizes our work and notes some difficulties of three-dimensional problems.

Chapter 2

Some Prior Work

This chapter reviews some computational work on triangulations. Materials covered are relevant to discussions in subsequent chapters. Further information on work done in the area can be found in the survey papers [Aure91, BeEp92] and the two books on computational geometry [Edel87, PrSh85]. A quick reference to this chapter is as follows.

1. The plane-sweep method computes an arbitrary triangulation for a plane geometric graph of n vertices in $\Theta(n \log n)$ time and linear storage (Section 2.1).

2. Dynamic programming is a general method to compute optimal polygon triangulations in polynomial time (Section 2.2).

3. The edge-flip scheme computes the Delaunay triangulation in quadratic time and linear storage. With some modifications, it serves as a popular heuristic for computing other optimal triangulations (Section 2.3).

4. $mst(S) \subseteq rng(S) \subseteq gg(S) \subseteq \mathcal{D}(S)$, where S is a point set, $mst(S)$ its minimum spanning tree, $rng(S)$ its relative neighborhood graph, $gg(S)$ its Gabriel graph, and $\mathcal{D}(S)$ its Delaunay triangulation (Section 2.4).

5. There is a subgraph approach to compute a triangulation of a point set so that its total edge length is a factor of $O(\log n)$ higher than that for the minimum length triangulation of the point set (Section 2.5).

2.1 Plane-Sweep

Plane-sweep is a simple way to construct an arbitrary triangulation for a plane geometric graph. Let us first see how to apply plane-sweep to a set of vertices $p_0, p_1, \ldots p_{n-1}$, sorted in this order from left to right. We conceptually sweep a vertical line from left to right, and maintain a triangulation for the vertices already encountered. When we reach vertex p_i, we add the edge $p_i p_{i-1}$. Then we march in a counterclockwise and then a clockwise direction along the convex hull of $p_0, p_1, \ldots, p_{i-1}$ to add edges between p_i and convex hull vertices. To perform this for the counterclockwise direction, we first set $p_j := p_{i-1}$ and let p_k be the convex hull vertex following p_j in this direction. If p_i, p_j and p_k form a right turn[1], we add $p_i p_k$, set $p_j := p_k$ and repeat the test; otherwise, we are done for p_i. The clockwise direction is done symmetrically.

The time spent to add an edge is constant since marching along convex hull edges in a quad-edge data structure takes constant time per step. Thus, a total of $O(n)$ time is needed to create $O(n)$ edges for the output. Sorting the n vertices (with x-coordinate as the primary key and y-coordinate as the secondary key) takes $\Omega(n \log n)$ time, which implies that plane-sweep takes time $O(n \log n)$ to compute an arbitrary triangulation for an unsorted set of n vertices. Although plane-sweep is a rather simple-minded technique, it constructs a triangulation in asymptotically optimal time. Indeed, $\Omega(n \log n)$ is necessary because triangulation is no easier than sorting [PrSh85, page 188].

Plane-sweep works equally well in time $O(n \log n)$ to construct a constrained triangulation for a plane geometric graph. In this case, we need a dictionary, for example, a splay tree, to record "active" (constraining line) segments intersecting the sweep line in sorted order. We also record the rightmost vertex within the region between two adjacent active segments (and above the topmost and below the bottommost active segment). The triangulation is constructed incrementally to the left of the sweep line. Each rightmost vertex recorded plays the role of p_{i-1} in the vertex set case. The extra effort not encountered for the vertex set case is $O(\log n)$ for

[1] Three vertices $p_i = (\pi_{i1}, \pi_{i2}), p_j = (\pi_{j1}, \pi_{j2})$ and $p_k = (\pi_{k1}, \pi_{k2})$ form a *right* turn when vertex p_k is on the right of the directed line $\vec{p_i p_j}$. This is the case when the determinant $\begin{vmatrix} \pi_{i1} & \pi_{i2} & 1 \\ \pi_{j1} & \pi_{j2} & 1 \\ \pi_{k1} & \pi_{k2} & 1 \end{vmatrix} < 0$.

each insertion, deletion, and search in the dictionary. The total effort is $O(n \log n)$, the same as before.

2.2 Dynamic Programming

Dynamic programming finds the optimal solution by systematically combining solutions of smaller problems. Let us see how the method computes a minimum length polygon triangulation, i.e., a polygon triangulation that minimizes the sum of edge lengths. Let P be the simple polygon with vertices $p_0, p_1, \ldots, p_{n-1}$. Let $L[i, i+j]$ be the total edge length of an optimal triangulation for the region P' bounded by $p_{i+j}p_i$ and the edges from p_ip_{i+1} through $p_{i+j-1}p_{i+j}$, where indices are modulo n. We compute $L[i, i+j]$ recursively as follows. Clearly, $p_{i+j}p_i$ is the only edge of P' possibly not in P. If $p_{i+j}p_i$ crosses the boundary of P, then P' is not a simple polygon and we thus set $L[i, i+j] := \infty$. Otherwise, $p_{i+j}p_i$ is either an edge or a diagonal of P. In either case, $p_{i+j}p_i$ is a side of some triangle in the optimal triangulation of P'. Such a triangle decomposes P' into two smaller (possibly empty) subpolygons of P', which must also be optimal in terms of total edge length; we, thus, have

$$L[i, i+j] := |p_{i+j}p_i| + \min_{k=1,2,\ldots,j-1} (L[i, i+k] + L[i+k, i+j]).$$

With the above, we can find $L[0, n-1]$ by three nested loops, varying j from 1 to $n-1$, i from 0 to $n-1$, and k from 1 to $j-1$, to first solve small problems and then combine their solutions to solve larger problems. The loop with k also checks whether $p_{i+j}p_i$ crosses the boundary of P by testing whether any edge $p_{i+k}p_{i+k+1}$ crosses[2] $p_{i+j}p_i$. To obtain the optimal triangulation for P, we record the k that is chosen to optimize each $L[i, i+j]$, and use it later to trace the triangulation. In total, the algorithm takes cubic time for the nested loops, and quadratic storage for keeping $L[i, j]$ and k [Gilb79, Klin80].

It is easy to see that the algorithm can be modified to compute other criteria. For example, we can compute a min-max angle triangulation for a simple polygon in the same amount of time and storage. Also, we can compute a triangulation that lexicographically minimizes the non-

[2] Two edges ac and bd cross iff $abcd$ is a convex polygon, i.e., abc, bcd, cda and dab are all left turns or all right turns.

increasing vector of largest angles of triangles, but the time- and storage-complexity in this case are a factor of n higher as we need to compare and store vectors of linear size. Incidentally, the dynamic programming approach is not necessarily the most efficient way to compute optimal triangulations; for instance, the min-max angle criterion can actually be computed in $O(n^2 \log n)$ time and linear storage (Section 4.1).

2.3 Edge-Flip

Edge-flip is a local optimization method that operates on two triangles whose union forms a convex polygon. It was used in [Laws72] to remove small angles: the edge bd shared by triangles abd and cbd is replaced or *flipped* when the smallest angle in these triangles is smaller than that of acb and acd. In effect, an edge-flip replaces two existing triangles by two new ones. This operation was incorporated into a plane-sweep scheme (Section 2.1) to incrementally compute a locally optimal triangulation $\mathcal{T}(S)$ [Laws77], that is, one that has no edge-flip to improve its quality. It was found that this locally optimality actually implies that $\mathcal{T}(S)$ is a completion of $\mathcal{D}(S)$ [Dela34, Sibs78]. Since any two completions of $\mathcal{D}(S)$ have the same value for their smallest angles, $\mathcal{T}(S)$ is actually a max-min angle triangulation (see [Edel87, page 301–303]).

There are various possibilities to implement the above scheme. One variant is to first compute an arbitrary triangulation, $\mathcal{T}(S)$, and then use a stack to schedule edge-flips. First, all edges of $\mathcal{T}(S)$ are pushed onto the stack. Then, edges are popped one by one to check for possible edge-flips. When an edge-flip bd occurs, each of the other four edges of the two triangles incident to bd is pushed onto the stack if it is not already there. The algorithm stops when the stack is empty, which occurs after at most $O(n^2)$ edge-flips where $n = |S|$; see, for example, [Edel92].

Many applications adapt the edge-flip method to optimize other criteria, by changing the condition for edge-flip. These, however, do not always compute the defined optimal triangulations and are thus heuristic schemes; see, for example, [DLR90a]. Also, it is unclear how long each heuristic runs since edges removed can be recycled back for further edge-flips. One possible way to cope with this difficulty is to impose a time limit on the algorithm; one other is to use a priority queue (instead of a stack) to schedule edge-flips and stop the algorithm once an unsuccessful edge-flip is encountered. When this latter heuristic is applied to the minimum length criterion (for which we flip long edges for short ones), it guarantees that at most $O(n^2)$ edges are flipped as removed edges do not reappear.

2.4 Delaunay Triangulations and Related Structures

The Delaunay triangulation has been a prominent subject in the study of triangulations; it is related to many positive results known in the area. Besides the $O(n^2)$ time edge-flip method (Section 2.3), Delaunay triangulation can be computed in $\Theta(n \log n)$ time by diverse algorithmic paradigms such as divide-and-conquer [GuSt85, ShHo75], geometric transformation [Brow79], and plane-sweep [Fort87]. Furthermore, it can be computed in time $O(n \log n)$ with high probability by randomized incrementation [GKS92]. Among all triangulations of a given point set S, the Delaunay triangulation optimizes criteria such as the max-min angle [Sibs78], the min-max circumscribed circle [D'AS89], the min-max smallest enclosing circle [D'AS89, Raja91], and the minimum integral of the gradient squared [Ripp90].

As a graph structure, the Delaunay triangulation of S, $\mathcal{D}(S)$, is the straight line dual of the so-called *Voronoi diagram* [Voro07, Voro08, Aure91]. Various subgraphs of $\mathcal{D}(S)$ have been studied in the literature. Three of those subgraphs, satisfying

$$mst(S) \subseteq rng(S) \subseteq gg(S) \subseteq \mathcal{D}(S)$$

will now be discussed.

<u>THE GABRIEL GRAPH OF S, $gg(S)$</u>. Given a vertex set S, $gg(S)$ is a unique graph that contains an edge pq, $p, q \in S$, if all points in $S - \{p, q\}$ are outside the closure of the disk with diameter pq. It is defined by Gabriel and Sokal [GaSo69] for use in geographical analysis, and, more generally, for clustering points. Obviously, an edge in $gg(S)$ is also a Delaunay edge, but the converse is not necessarily true; thus, $gg(S) \subseteq \mathcal{D}(S)$.

<u>THE RELATIVE NEIGHBORHOOD GRAPH OF S, $rng(S)$</u>. An edge pq belongs to $rng(S)$ if

$$|pq| \leq \min_{x \in S - \{p,q\}} \max\{|xp|, |xq|\}.$$

This definition goes back to Toussaint [Tous80], who modified a similar definition by Lankford [Lank69], for use in pattern recognition. Equivalently, the *lune* of pq, denoted by λ_{pq}, is the set $\{x \in \mathbb{R}^2 : \max\{|xp|, |xq|\} < |pq|\}$, and pq in $rng(S)$ if $\lambda_{pq} \cap S = \emptyset$. It is obvious that $\lambda_{pq} \cup \{p, q\}$ contains the closure of the disk with diameter pq; thus, each edge of $rng(S)$ is also an edge of $gg(S)$. Therefore, $rng(S) \subseteq gg(S)$.

A MINIMUM SPANNING TREE OF S, $mst(S)$. It is a spanning tree, i.e., a connected and cycle free graph, on S with the minimum total edge length. From the definition, the lune of each edge pq in $mst(S)$ must be empty. Otherwise, there is a point $r \in \lambda_{pq}$ so that pr and qr are not both in $mst(S)$, as $mst(S)$ is cycle free. The spanning tree obtained by replacing pq with either pr or qr has smaller total edge length than $mst(S)$, a contradiction. Thus, we have $mst(S) \subseteq rng(S)$.

All the graphs, $mst(S), rng(S), gg(S)$, and $\mathcal{D}(S)$ are connected since $mst(S)$ is connected, and are plane since $\mathcal{D}(S)$ is plane. An efficient way to compute $mst(S)$, $rng(S)$ or $gg(S)$ is to first compute $\mathcal{D}(S)$ with some of the methods mentioned above, and then remove invalid edges; see, for example, [MaSo80, Supo83, Yao82].

2.5 Minimum Length Triangulations

We have seen in Section 2.2 the cubic time (dynamic programming) algorithm for constructing minimum length triangulation for a simple polygon. The same problem for a point set is open (Section 1.3.2, Problem 5) [Lloy77]. We mention in the following an interesting attempt, termed *subgraph approach*, for finding a polynomial time solution to the problem.

The idea is to convert a point set problem to a number of (simple) polygon problems, which can then be solved in polynomial time, for example, by dynamic programming. This conversion is done by adding sufficiently many non-intersecting edges to the point set so as to divide its convex hull into a number of polygons. For this to work, we must assert that these additional edges are indeed part of an optimum and are computable in polynomial time. Unfortunately, such a set of edges has not been found. Currently, only a small subset of edges can be identified (see [Gilb79] for details).

Despite the above difficulty, the subgraph approach was used as a powerful technique by Lingas [Ling87] and Plaisted and Hong [PlHo87, OTZ88] to compute a triangulation with a guaranteed bound on its total edge length. The former uses the union of the convex hull and a spanning forest to reduce a given point set problem to several polygon problems. Under the assumption of a uniform point distribution, the result is a triangulation with total edge length within a factor of $O(\log n)$ from the optimum with high probability, and the expected length of the solution is of the same order as that of the optimum. The latter generates polygon problems by computing the convex hull and a star graph from each point to its nearby neighbors. After optimally triangulating all polygons, this method generates a solution with total edge length within a factor of $O(\log n)$ from the optimum.

Chapter 3

The Edge Insertion Paradigm

This chapter introduces an operation called the *edge-insertion* to locally improve a triangulation. Basically, the operation adds a new edge to the triangulation to be improved, deletes old edges intersecting the new edge, and retriangulates the resulting polygons supported by the new edge. This is in some sense a generalization of the edge-flip operation (Section 2.3)—an edge-insertion generally intersects many old edges, but is the same as an edge-flip when it intersects only one. Its ability to intersect many edges turns out to be very powerful; its formalism into an iterative improvement scheme, termed the *edge-insertion paradigm*, computes a few min-max and max-min optimal triangulations for which edge-flip paradigm fails, as shown in the next chapter.

In the following, we examine an abstraction of the edge-insertion paradigm based on two sufficient conditions for the min-max (and max-min) criteria that it can optimize. Section 3.1 formalizes the basic version of the paradigm, and Section 3.2 states the two sufficient conditions. Then, Section 3.3 proves the correctness of the paradigm when applied to such criteria. Section 3.4 discusses refinements to the paradigm, Section 3.5 presents some extensions, and Section 3.6 summarizes this chapter.

3.1 The Paradigm

Let S be a point set in $\mathbb{R}^2$, and let x, y, z be points in S. Recall that xy denotes the line segment that connects x and y, and xyz denotes the triangle with vertices x, y, and z. We call xyz an *empty* triangle if all other points of S lie outside the closure of xyz.

A *measure* μ is a function that maps each triangle xyz to a real value $\mu(xyz)$. Examples of measures that are of particular interest here are largest angle, height, slope, and eccentricity of a triangle. We restrict our attention to min-max criteria, that is, for each μ we consider the construction of a triangulation of S whose maximum $\mu(xyz)$ over all its triangles is the smallest among all possible triangulations of S. Max-min criteria can be simulated by considering $-\mu$.

The *measure* of a triangulation $\mathcal{A}$ is defined as $\mu(\mathcal{A}) = \max\{\mu(xyz) : xyz \text{ a triangle of } \mathcal{A}\}$. If $\mathcal{A}$ and $\mathcal{B}$ are two triangulations of a common point set, then $\mathcal{B}$ is called an *improvement* of $\mathcal{A}$, denoted by $\mathcal{B} \prec \mathcal{A}$, if $\mu(\mathcal{B}) < \mu(\mathcal{A})$, or $\mu(\mathcal{B}) = \mu(\mathcal{A})$ and the set of triangles xyz in $\mathcal{B}$ with $\mu(xyz) = \mu(\mathcal{B})$ is a proper subset of the set of such triangles in $\mathcal{A}$. A triangulation $\mathcal{A}$ is *optimal* for μ if there is no improvement of $\mathcal{A}$.

The formal specification of the *edge-insertion* of qs, $q, s \in S$, into a triangulation $\mathcal{A}$ of S is as follows.

Function EDGE-INSERTION($\mathcal{A}$, qs): triangulation.
1. $\mathcal{B} := \mathcal{A}$.
2. Add qs to $\mathcal{B}$ and remove from $\mathcal{B}$ all edges that intersect qs.
3. Retriangulate polygons P and R constructed in step 2.
4. **return** $\mathcal{B}$.

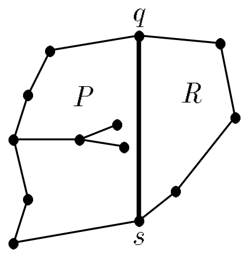

Figure 3.1: P and R are created by the removal of edges intersecting qs.

In step 2, P and R created are not necessarily simple polygons in the usual meaning of the term, as shown by Figure 3.1. Although their interiors are always simply connected, there can be edges (or "cracks") contained in the interiors of their closures. Nevertheless, as discussed in Section 1.2, each such edge can be treated as if it consisted of two edges, one for each side, which then allows us to treat P and R as if they were simple polygons. As such, we can triangulate P and R in step 3 in an optimal fashion (minimizing the maximum μ) by, for example, dynamic programming (Section 2.2).

With the edge-insertion operation, we formulate the most basic version of the *edge-insertion paradigm* as follows; it tries all possible edge-insertions and halts when no edge-insertion improves the current triangulation.

Input. A set S of n points in $\mathbb{R}^2$.

Output. An optimal triangulation $\mathcal{T}$ of S.

Algorithm. Construct an arbitrary triangulation $\mathcal{A}$ of S;
repeat
  $\mathcal{T} := \mathcal{A}$;
  for all pairs $q, s \in S$ **do**
    $\mathcal{B} :=$ EDGE-INSERTION$(\mathcal{A}, qs)$;
    if $\mathcal{B} \prec \mathcal{A}$ **then** $\mathcal{A} := \mathcal{B}$; **exit** the for-loop **endif**
  endfor
until $\mathcal{T} = \mathcal{A}$.

To compute an optimal triangulation, this paradigm assumes that there is an edge-insertion that improves $\mathcal{A}$ whenever $\mathcal{A}$ is not yet optimal. Section 3.2 presents two conditions on measures for which the assumption is true, and Section 3.3 proves this assertion. Assuming this, we argue that the above algorithm runs in time $O(n^8)$. A single edge-insertion operation takes time $O(n^3)$ when retriangulating by dynamic programming, as long as the measures of any two triangles can be compared in constant time. Thus, the for-loop takes time $O(n^5)$ per iteration of the repeat-loop. Finally, the repeat-loop is iterated at most $O(n^3)$ times because there are only $\binom{n}{3}$ triangles spanned by S, and each iteration permanently discards at least one of them when it finds an improvement of the current triangulation.

3.2 Two Sufficient Conditions

Again, let S be a set of n points in $\mathbb{R}^2$. For each triangle xyz, $x, y, z \in S$, we designate one or more vertices as *anchors* of xyz, depending solely on the measure μ. For all but one of the measures of interest in the next chapter, vertices with the largest angle are anchors. In these cases, each non-isosceles triangle has a unique anchor.

Let $\mathcal{T}$ be a triangulation of S, and let xyz be an empty triangle of S. We say that $\mathcal{T}$ *breaks xyz at y* if it contains an edge yt with $yt \cap xz \neq \emptyset$. Note that if $\mathcal{T}$ breaks xyz at y, then it can neither break it at x nor at z since $\mathcal{T}$ is a plane graph.

We are now ready to formulate two conditions for measures μ. They both suffice to show that the edge-insertion paradigm computes a global optimum (i.e., min-max μ). The first condition requires that every improvement of a triangulation $\mathcal{A}$ of S has an edge that breaks one worst triangle of $\mathcal{A}$ at its (unique) anchor. Precisely, we require that

for every triangle xyz of $\mathcal{A}$, if $\mathcal{T}$ neither contains xyz nor breaks it at its anchor(s), then $\max\{\mu(\mathcal{A}), \mu(\mathcal{T})\} > \mu(xyz)$. (I)

The second condition is a stronger version of (I). It further requires that $\mu(\mathcal{T})$ is no better than measures of those triangles in $\mathcal{A}$ which $\mathcal{T}$ does not break at their anchors. Specifically, we require that

for every triangle xyz of $\mathcal{A}$, if $\mathcal{T}$ neither contains xyz nor breaks it at its anchor(s), then $\mu(\mathcal{T}) > \mu(xyz)$. (II)

We will see in Sections 4.3 and 4.4 that the slope and the eccentricity satisfy (I) but not (II). This shows that (I) is strictly weaker than (II). Because of the greater generality of (I), we can only derive an $O(n^3)$ time refinement to the edge-insertion paradigm, as compared to $O(n^2 \log n)$ time refinement for (II).

3.3 The Cake Cutting Lemma

This section shows that if $\mathcal{A}$ is not yet optimal for measure μ satisfying condition (I), then there is an edge whose insertion leads to an improvement, specifically an edge breaking a worst triangle at its anchor. This is achieved through the following two lemmas. The argument uses only (I), so also applies when (II) is satisfied.

Cake-Cutting Lemma. Assume μ satisfies condition (I). Let $\mathcal{T} \prec \mathcal{A}$ be two triangulations of a point set S; let pqr be a triangle in $\mathcal{A}$ but not in $\mathcal{T}$ with $\mu(pqr) = \mu(\mathcal{A})$; let q be an anchor of pqr; and let qs be an edge in $\mathcal{T}$ that intersects pr. Let P and R be the polygons generated by adding qs to $\mathcal{A}$ and removing all edges that intersect qs. Then there are triangulations $\mathcal{P}$ and $\mathcal{R}$ of P and R with $\mu(pqr) > \mu(\mathcal{P})$ and $\mu(pqr) > \mu(\mathcal{R})$.

Proof. We focus on triangulating P; R is triangulated similarly. Imagine we have P and $\mathcal{T}$ on separate pieces of transparent paper that we lay on top of each other so that the vertices of P match those corresponding ones in $\mathcal{T}$. Next, we clip everything of $\mathcal{T}$ outside P. Now, we see generally many edges of $\mathcal{T}$ cutting through P, but none of them can meet qs as $qs \in \mathcal{T}$. Let us call each connected component of an edge intersected with the interior of P a *clipped edge*. Since P is not necessarily convex, several clipped edges can belong to the same edge of $\mathcal{T}$. Our plan is to use these clipped edges as guides to successively remove ears from P to obtain a required $\mathcal{P}$.

If no clipped edge exists, then P has only three vertices and therefore must be a triangle of $\mathcal{T}$. Because this triangle is not in $\mathcal{A}$ and $\mathcal{T} \prec \mathcal{A}$, the triangle has measure less than $\mu(\mathcal{A})$. So we are done. In the following, we thus assume the existence of at least one clipped edge. Denote by $q = p_0, p_1, \ldots, p_k, p_{k+1} = s$ the sequence of vertices of P.

Claim 1. For $1 \le j \le k$, if $\angle p_{j-1}p_jp_{j+1} < \pi$, then $p_{j-1}p_{j+1}$ is a diagonal of P.

Proof (of Claim 1). By construction of P, it is possible to find non-intersecting line segments $p_{j-1}x$ and $p_{j+1}y$, both inside P, so that x and y lie on qs. (If $j = 1$, then $x = p_{j-1} = q$; if $j = k$, then $y = p_{j+1} = s$.) The (possibly degenerate) pentagon $xp_{j-1}p_jp_{j+1}y$ is part of P, and because the interior angles at p_j, x, and y measure less than π, edge $p_{j-1}p_{j+1}$ is a diagonal of the pentagon and therefore also of P. This completes the proof of Claim 1.

A clipped edge divides P into two polygons, the *near side* supported by qs and the *far side* not supported by qs.

Claim 2. There is at least one clipped edge whose far side is a triangle.

Proof (of Claim 2). Let xy be a clipped edge so that its far side, F, contains no further clipped edge. Let ab be the edge in $\mathcal{T}$ that contains xy, and let abc be the triangle in $\mathcal{T}$ that lies on the same side of the line through xy as F. We have $F \subseteq abc$ for otherwise F contains a clipped edge belonging to ac or bc. Also, all vertices of F, except possibly x and y, are points in S and therefore equal to a, b, or c. Thus, F is a required triangle. This proves Claim 2.

The clipped edges xy that satisfy Claim 2 fall into four classes as illustrated in Figure 3.2. An ear $p_{i-1}p_ip_{i+1}$ so that xy is a clipped edge with far side xp_iy can now be removed from

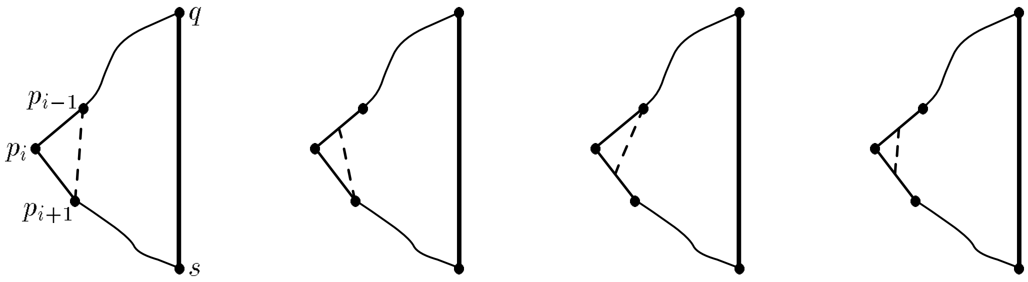

Figure 3.2: A clipped edge xy that satisfies Claim 2 has zero, one, or two endpoints on edges of P.

P, leaving a polygon P' with one less vertex. Claims 1 and 2 remain true for P' because the removed ear is not supported by qs. So we can iterate and compute a triangulation $\mathcal{P}$ of P. Symmetrically, we get a triangulation $\mathcal{R}$ of R. Let $\mathcal{B}$ be the thus obtained triangulation of S.

Claim 3. $\mu(pqr) > \mu(abc)$ for all triangles abc in $\mathcal{P}$ and $\mathcal{R}$.

Proof (of Claim 3). It suffices to show the claim for triangle abc in $\mathcal{P}$ and $\mathcal{R}$ with maximum μ. Assume without loss of generality that abc is in $\mathcal{P}$ and $\mu(abc) = \mu(\mathcal{B})$. At the time immediately before abc was removed by adding ac, there was a clipped edge xy with far side xby. Hence, $\mathcal{T}$ does not break abc at b, and by construction, $\mathcal{A}$ breaks abc at b and therefore neither at a nor at c.

If $xy = ac$, then abc is a triangle in $\mathcal{T}$ but not in $\mathcal{A}$, and therefore $\mu(pqr) > \mu(abc)$. So, assume $xy \neq ac$. Then, if b is an anchor of abc, (I) implies that $\max\{\mu(\mathcal{B}), \mu(\mathcal{T})\} > \mu(abc)$ as $\mathcal{T}$ does not break abc at b nor does it contain abc. This simply means that $\mu(\mathcal{T}) > \mu(abc)$ because $\mu(\mathcal{B}) \not> \mu(abc)$, and therefore $\mu(pqr) > \mu(abc)$. Otherwise, a or c is an anchor. As in the previous case only with $\mathcal{T}$ replaced by $\mathcal{A}$, we obtain $\max\{\mu(\mathcal{B}), \mu(\mathcal{A})\} > \mu(abc)$ from (I) and thus $\mu(pqr) = \mu(\mathcal{A}) > \mu(abc)$. This completes the proof of Claim 3 and also the Cake-Cutting Lemma. $\square$

The Cake-Cutting Lemma now shows that the basic edge-insertion paradigm cannot get stuck in a local optimum for μ satisfying condition (I).

Lemma 3.1 Assume μ satisfies condition (I). Let $\mathcal{A}$ be a non-optimal triangulation of a point set S. Then there is an edge-insertion operation that improves $\mathcal{A}$.

Proof. Let $\mathcal{B}$ be an improvement of $\mathcal{A}$ and consider a triangle pqr in $\mathcal{A}$ with $\mu(pqr) = \mu(\mathcal{A})$ that is not in $\mathcal{B}$. Assuming q is an anchor of pqr, condition (I) implies that $\mathcal{B}$ contains an edge qs with $qs \cap pr \neq \emptyset$. Let P and R be the polygons generated by adding qs and deleting the edges that intersect qs. The Cake-Cutting Lamma implies that there are polygon triangulations $\mathcal{P}$ and $\mathcal{R}$ of P and R with $\mu(\mathcal{P})$ and $\mu(\mathcal{R})$ both smaller than $\mu(pqr)$. $\square$

3.4 Refinements of the Paradigm

We now refine the basic paradigm for measures satisfying (I) and (II). Both refinements are specializations of the algorithm given below. It differs from the basic paradigm in two major ways. First, edge-insertions are restricted to *candidate edges* qs that break a worst triangle pqr at its anchor q. These edges are tried in a sequence, say $qs_1, qs_2, \ldots$, etc. Second, the two polygons created by adding qs are retriangulated by repeatedly removing ears (as in the proof of the Cake-Cutting Lemma), rather than by dynamic programming. We use the notation $s_{i+1} = \text{NEXT}(s_i)$.

Input. A set S of n points in $\mathbb{R}^2$.

Output. An optimal triangulation $\mathcal{T}$ of S.

Algorithm. Construct an arbitrary triangulation $\mathcal{A}$ of S;
repeat
  $\mathcal{T} := \mathcal{A}$;
  find a worst triangle pqr in $\mathcal{A}$, let q be its anchor, and set $s := s_1$;
  while s is defined **do**
    $\mathcal{B} := \mathcal{A}$, add qs to $\mathcal{B}$, and remove all edges that intersect qs;
    (partially) triangulate the two polygons P and R
      by cutting off ears xyz with $\mu(xyz) < \mu(pqr)$;
    if P and R are completely triangulated **then**
      $\mathcal{A} := \mathcal{B}$; **exit** the while-loop
     else $s :=$ NEXT(s)
    endif
  endwhile
until $\mathcal{T} = \mathcal{A}$.

Two remarks are in order. First, this algorithm finds a triangulation with min-max triangle measure, but not necessarily an optimal triangulation in the sense that the set of worst triangles is minimal. To achieve this slightly more ambitious goal, the repeat-loop must not halt until all worst triangles have subjected to unsuccessful edge-insertions. This requires only minor modification to the algorithm. Second, in an implementation of the algorithm we would not really copy an entire triangulation. Instead of the assignment $\mathcal{T} := \mathcal{A}$, we would use a flag to check whether an iteration of the repeat-loop produced an improvement. And, the assignment of $\mathcal{B} := \mathcal{A}$ and the subsequent $\mathcal{A} := \mathcal{B}$ can be avoided by making changes directly in $\mathcal{A}$ and undoing them to the extent necessary.

The following subsections explain some of the steps in greater details and analyze the time- and storage-complexity of the two refinements.

3.4.1 Triangulating by Ear Cutting — Part 1

As implied by the proof of the Cake-Cutting Lemma, the sequence in which ears are removed from P is immaterial so long as only the last is supported by qs. Additionally, three consecutive vertices is an ear of the remaining portion of P yet to be triangulated if they form an angle less than π inside P. This condition is equivalent to they form a left turn when vertices of P are listed in a counterclockwise order. Thus, the triangulation of P can be implemented using a stack and with simple test of an ear, so that it runs in time linear in the size of P. Here are the details. We first initialize the stack, say *stackP*, by pushing q followed by p. After that, we consider the other vertices of P one by one in order. Each time a vertex p_i is considered, we first reset the *stop* indicator, repeat the CUTEARP(p_i) procedure until *stop* is set or *stackP* has only one vertex, then push p_i onto *stackP*.

procedure CUTEARP(s)
  suppose p_k is the topmost vertex on *stackP* followed by p_{k-1}.
  if $p_{k-1}p_k s$ is an ear of the remaining portion of P yet to be triangulated
    and $\mu(p_{k-1}p_k s) < \mu(pqr)$ **then**
      use $p_{k-1}p_k s$ in triangulating P;
      pop p_k from *stackP*
    else
      set *stop* := **true**
  endif.

The triangulation is complete if, at the end of the process, there are only two vertices on *stackP*. Similarly, we use *stackR* and procedure CUTEARR(s) to triangulate R.

3.4.2 Analysis under (I)

The above ear-cutting process speeds up the algorithm for measures satisfying (I) and (II). But, we can do better for (II) by further refinement. This is because partial effort on unsuccessfully triangulating P and R can be saved for subsequent attempts. This results in further speed up when integrated into a clever way to search for a good edge-insertion. We will return to this discussion after we analyze the current refinement.

Theorem 3.2 Let S be a set of n points in $\mathbb{R}^2$ and let μ be a measure that satisfies (I). A triangulation of S that minimizes the maximum μ can be constructed in time $O(n^3)$ and storage $O(n^2)$.

Proof. To achieve the claimed bounds, we use the algorithm above, along with two data structures requiring a total of $O(n^2)$ storage. First, the quad-edge data structure (Section 1.2) stores the triangulation in $O(n)$ storage and admits common operations, such as removing an edge, adding an edge, and walking from one edge to the next in constant time each.

Second, to record the status of candidate edges, we use an n-by-n bit array whose elements correspond to the edges defined by S. If the insertion of a candidate edge qs is unsuccessful, that is, the triangulation of P or R cannot be completed, then we know by the Cake-Cutting Lemma that qs cannot be in any improvement of the current triangulation. We then set the bit for qs, so that we do not attempt the insertion of qs again. If the insertion of qs is successful, we set the bit for pr since it cannot be in any later improvement (as implied by (I) that every improvement breaks pr). The bit array is also used to compute the sequence of candidate edges: scan the row corresponding to q and take all edges qs that intersect pr and whose flags have not yet been set.

Each edge-insertion, whether successful or not, causes a new flag set for one of the $\binom{n}{2}$ edges defined by S. Therefore, at most $\binom{n}{2}$ edge-insertions are carried out, taking a total of $O(n^3)$ time. The claim follows because an initial triangulation can be constructed in time $O(n \log n)$, most straightforwardly by plane-sweep (Section 2.1). $\square$

3.4.3 Triangulating by Ear Cutting — Part 2

We now continue to refine the ear cutting process to triangulate P and R for measures satisfying (II). To be accurate, we should mention that the task of cutting ears and that of searching for a candidate edge are woven together. The complete algorithm is given in the next subsection.

The refinement centers at saving partial work done in an unsuccessful edge-insertion of qs_{i-1} to qs_i. In particular, some removed ears due to qs_{i-1} may remain valid for $qs_i = qs$. Thus, we just need to restore P and R to the extent necessary and then triangulate their remaining portions on their stacks. For reason of efficiency (Section 3.4.5), we alternate removing an ear from each, and when one polygon is successfully triangulated, we attempt to complete the polygon that remains. This is formalized in the following CutEars procedure.

```
procedure CutEars
    stop := false;
    while stackP and stackR each has ≥ 2 vertices and not stop do
      cutearP(s); if not stop then cutearR(s) endif
    endwhile;
    while stackP has ≥ 2 vertices and not stop do cutearP(s) endwhile;
    while stackR has ≥ 2 vertices and not stop do cutearR(s) endwhile.
```

If the procedure finishes without raising the flag (*stop* = **false**), then we must have only one vertex on each stack, and thus the triangulations for P and R are complete and an improvement has been obtained. Otherwise, the flag is raised while testing either P or R (so we should really have used two flags to be able to distinguish the two cases—we pretend we did).

Let us now consider the case where the triangulation of P cannot be completed due to the insertion of candidate edge qs. Let P' be the portion of P defined by vertices $q = p_0, p_1, \ldots, p_k$ on *stackP* and $s = p_{k+1}$. By construction, each possible ear $p_{j-1}p_jp_{j+1}$, $j \leq k$ of P' is such that $\mu(p_{j-1}p_jp_{j+1}) > \mu(pqr)$. The next lemma is crucial for defining NEXT(s).

Lemma 3.3 Let $\mathcal{B}$ be an improvement of $\mathcal{A}$ for μ satisfying condition (II). Then all edges of $\mathcal{B}$ that intersect the interior of P' also intersect qs. In particular, all edges of $\mathcal{B}$ incident to q avoid the interior of P'.

Proof. As in the proof of the Cake-Cutting Lemma, we consider P' as a "window" through which we see clipped edges of $\mathcal{B}$. Now suppose the claim is not true, that is, there is at least one clipped edge with no endpoints on qs. As before we thus find such a clipped edge xy whose far side is a triangle xp_jy. But now condition (II) implies $\mu(\mathcal{B}) > \mu(p_{j-1}p_jp_{j+1})$ if p_j is an anchor of the ear $p_{j-1}p_jp_{j+1}$, and $\mu(\mathcal{A}) > \mu(p_{j-1}p_jp_{j+1})$ if p_{j-1} or p_{j+1} is an anchor. This contradicts the assumption that P' has no such ear. $\square$

It is interesting to observe that the proof of Lemma 3.3 breaks down if we assume that μ satisfies only (I). Symmetrical result holds for R', the portion of R defined by vertices $q = r_0, r_1, \ldots, r_m$ on *stackR* and $s = r_{m+1}$, when R cannot be completed due to the insertion of candidate edge qs.

3.4.4 Searching for a Proper Edge

As we search for an edge-insertion, we maintain an open wedge W containing all the remaining candidate edges. Initially, W is the wedge between the ray $\vec{qp}$ and the ray $\vec{qr}$. If the edge-insertion of qs_i turns out to be unsuccessful because the triangulation of P cannot be completed, then Lemma 3.3 allows us to refine W as the part of the old W on R's side of $\vec{qs_i}$; see Figure 3.3. Similarly, if the triangulation of R cannot be completed, then W can be narrowed down to

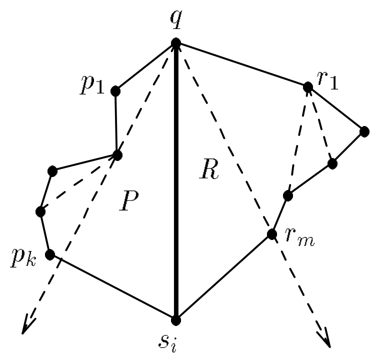

Figure 3.3: The two rays define the current W, and the broken line segments indicate those triangles removed from P and R. If P is found to be noncompletable, then the next candidate edge qs_{i+1} lies in the updated W defined by $\vec{qs_i}$ and the ray passing through R.

P's side of $\vec{qs_i}$. (As a consequence, if neither P nor R can be completed, then it is impossible to improve the current triangulation by breaking pqr at q. This, however, turns out to be too costly to check; see Section 3.4.5.)

Assume that qs_i has failed because P could not be completed. Because qs_{i+1} intersects $r_m s_i$, and thus moves away from P', all ears cut off P' remain the same and do not have to be reconsidered. On the other hand, s_i is no longer a vertex of R, so all ears cut off R' that are incident to s_i must be returned to the territory of R'.

We can now integrate the tasks of searching for a good edge-insertion and triangulating P and R. This is shown in the next procedure. The searching is done by stepping from triangle to triangle. The first vertex s that we test is the third vertex of the other triangle of pr (if no such triangle exists, then pr is an edge of the convex hull of S and no appropriate vertex s exists). In general, each subsequent s is just the third vertex of the triangle incident to $p_k r_m$ where s and q lie on opposite sides of the line through $p_k r_m$. This vertex is denoted by THIRD(p_k, r_m). For each new vertex s obtained in this fashion, we distinguish two cases: if $s \in W$, then an edge-insertion is performed; else, we remove ears as much as possible. Note that W is defined by $\vec{qp}_w$ and $\vec{qr}_w$.

Input. A triangulation $\mathcal{A}$ of S with triangle pqr so that $\mu(pqr) = \mu(\mathcal{A})$.

Output. An improved triangulation or a message that the measure cannot be improved.

Algorithm. Set $p_w := p$, $r_w := r$; **push** q then p onto *stackP*, and q then r onto *stackR*;
loop
  if THIRD(p_k, r_m) is not defined **then**
    return the message that the measure cannot be improved and **stop**
  else
    set $s :=$ THIRD(p_k, r_m), and remove $p_k r_m$ from $\mathcal{A}$;
    if $s \in W$ **then**
      add qs to $\mathcal{A}$ and attempt the triangulation of P' and R' by CutEars
      case 1: The attempt succeeds. Return the new triangulation and **stop**;
      case 2: The flag was raised while testing P.
        Set $p_w := s$, and **push** s onto *stackP*;
        restore ears cut off R' that are incident to s;
      case 3: The flag was raised while testing R.
        Set $r_w := s$, and **push** s onto *stackR*;
        restore ears cut off P' that are incident to s
    else (i.e., $s \notin W$)
      if sr_m intersects W **then**
        stop := **false**; **while not** *stop* **do** CUTEARP(s) **endwhile**;
        push s onto *stackP*
      else (i.e., sp_k intersects W)
        stop := **false**; **while not** *stop* **do** CUTEARR(s) **endwhile**;
        push s onto *stackR*
      endif
    endif
  endif
forever.

3.4.5 Analysis under (II)

Because of the alternation between removing an ear from P' and one from R', at most only one more than half of the removed ears are restored. This is also true if one polygon is completely triangulated while ears are still removed from the other, because in this case only the ears of the former polygon need to be restored, and their number is smaller than those cut off from the other. Thus, the total number of removed ears while edge-inserting $qs_1, qs_2, \ldots, qs_l$ is linear in the number of old edges intersected by qs_l, and so does the running time of the procedure.

We next prove that the old edges removed will never be reinserted in any later successful edge-insertion, which then implies the claimed time-complexity of the algorithm.

Lemma 3.4 Let $\mathcal{A}$ be a triangulation of S, with worst triangle pqr, and let $\mathcal{B}$ be obtained from $\mathcal{A}$ by the successful insertion of an edge qs_l. Then no edge xy in $\mathcal{A}$ that intersects qs_l can be an edge of any improvement of $\mathcal{B}$.

Proof. Lemma 3.3 implies that every improvement of $\mathcal{B}$ has an edge qw that lies inside the wedge W computed when qs_l is inserted into $\mathcal{A}$. Every edge xy in $\mathcal{A}$ that intersects qs_l also intersects every other edge qt with $t \in W$. In particular, $xy \cap qw \neq \emptyset$ which implies that xy is neither in $\mathcal{B}$ nor in any improvement of $\mathcal{B}$. $\square$

Theorem 3.5 Let S be a set of n points in $\mathbb{R}^2$, and let μ be a measure that satisfies (II). A triangulation of S that minimizes the maximum μ can be constructed in time $O(n^2 \log n)$ and storage $O(n)$.

Proof. As before, the algorithm uses the quad-edge data structure to store the triangulation. The bit array, however, is replaced by a priority queue that holds the triangles of $\mathcal{A}$ ordered by measure. It admits inserting and deleting triangles and finding a triangle with maximum measure in logarithmic time each [CLR90]. Lemma 3.4 implies that only $O(n^2)$ edges and triangles are manipulated in the main loop of the algorithm, which thus takes time $O(n^2 \log n)$. Lemma 3.4 also implies a quadratic upper bound on the number of iterations of the repeat-loop, which implies that the total time needed to find worst triangles pqr is also $O(n^2 \log n)$. $\square$

3.5 Extensions

In this section, we address the extensions to constrained problem and to problem defined by vector of measures. The Cake-Cutting Lemmas and Lemma 3.1 remain valid for constrained triangulations provided the measures satisfies (I) or (II) also in this more general setting. In this case, we only need to modify the algorithm so that edges intersecting constraining edges can never be used as candidate edges for insertions. This modification does not increase the time-complexity.

It follows that the unique triangulation that lexicographically minimizes the decreasing vector of triangle measures can be constructed for the non-degenerate case where no two triangles of the point set have the same measure. First, construct a min-max triangulation, $\mathcal{T}_1$, and declare the three edges of the triangle with the largest measure as constraining edges. Second, construct a min-max triangulation $\mathcal{T}_2$ for the thus constrained input and introduce new constraints to enforce the second largest measure in future triangulations. Continue this way and construct triangulations $\mathcal{T}_3$, $\mathcal{T}_4$ and so on, until the constraining edges add up to a triangulation themselves.

Interestingly, the time- and storage-complexity remain the same. As before, for measures satisfying (I), it is because each edge needs to be inserted at most once during the entire process. For measures satisfying (II), each edge once removed cannot reappear in any future triangulations. We summarize the results as follows.

Corollary 3.6 Let S be a set of n points in $\mathbb{R}^2$, with or without constraining edges. Let μ be a measure that satisfies (I).

(1) A triangulation of S that minimizes the maximum μ can be constructed in time $O(n^3)$ and storage $O(n^2)$.

(2) In the non-degenerate case, when $\mu(xyz) \neq \mu(abc)$ unless $xyz = abc$, the triangulation that lexicographically minimizes the decreasing vector of measures μ can be constructed in the same amount of time and storage.

Corollary 3.7 Let S be a set of n points in $\mathbb{R}^2$, with or without constraining edges. Let μ be a measure that satisfies (II).

(1) A triangulation of S that minimizes the maximum μ can be constructed in time $O(n^2 \log n)$ and storage $O(n)$.

(2) In the non-degenerate case, the triangulation that lexicographically minimizes the decreasing vector of measures μ can be constructed in the same amount of time and storage.

3.6 Discussion

The main result of this chapter is the formulation of the edge-insertion paradigm as a general method to compute optimal triangulations, and the identification of two classes of criteria for which the paradigm indeed finds an optimum. The algorithm for measures satisfying (I) appears in Sections 3.4 and 3.4.1, and for (II) in Sections 3.4, 3.4.3, and 3.4.4.

Though usually simple to verify, conditions (I) and (II) are somewhat restrictive. Currently, only four measures are known to satisfy them (next chapter). It would be interesting to find conditions weaker than (I) even though the price to pay may be implementations of the paradigm that take more than cubic time. On the other hand, it remains open whether further speed up is possible for the two refinements presented.

Another problem suggested in Section 3.5 is how to optimize vectors of measures in the degenerate case where multiple measures occur. We note that the special case of this problem for a simple polygon can be handled easily by dynamic programming in time $O(n^4)$ and storage $O(n^3)$.

Chapter 4

Applications of Edge Insertion

This chapter shows four applications of the edge-insertion paradigm. The main result is the following theorem derived from corollaries 3.6 and 3.7.

Theorem 4.1 For a set of n vertices S in $\mathbb{R}^2$, with or without constraining edges,

(1) a min-max angle triangulation can be computed in time $O(n^2 \log n)$ and storage $O(n)$,

(2) a max-min height triangulation can be computed in time $O(n^2 \log n)$ and storage $O(n)$,

(3) a min-max slope triangulation can be computed in time $O(n^3)$ and storage $O(n^2)$, and

(4) a min-max eccentricity triangulation can be computed in time $O(n^3)$ and storage $O(n^2)$.

In addition, the generalization of each criterion to the vector form can be computed in the same amount of time and storage, provided the input is non-degenerate.

To prove Theorem 4.1, we show that the measure μ satisfies condition (II) when defined as the largest angle or the (negative) height (Sections 4.1 and 4.2), and it satisfies (I) when defined as the slope or the eccentricity (Sections 4.3 and 4.4). In each case, the proof also holds under the more general setting with constraining edges. For these to be a worthy exposition, we must first be convinced that other known methods, especially the edge-flip scheme, fail to optimize these measures. In addition, we must ascertain that these criteria, plus the empty disk property, do not necessarily define the same optima. These issues are discussed in the next few paragraphs.

Recall that the edge-flip scheme iteratively flips an edge as long as it makes a local improvement. As each flip is allowed only for two abutting triangles forming a convex quadrilateral, it is highly probable that the method gets stuck in local optima. For instance, consider the triangulation of Figure 4.1 when subject to edge-flip under min-max angle, max-min height, or min-max eccentricity. It is a regular pentagon $abcde$ that was slightly perturbed. More

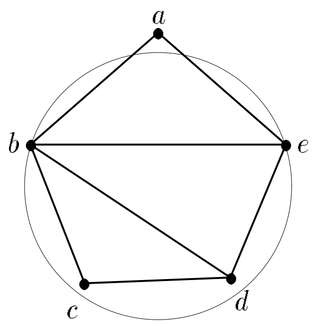

Figure 4.1: Flipping be or bd cannot locally improve this triangulation. Thus, the edge-flip scheme cannot change the shown triangulation into the optimal one containing edges ac and ad.

precisely, the perturbation is such that $|bc| > |de|$, $\angle abc = \angle aed < \angle bcd < \angle cde$, and also $h(d, ce) < h(c, bd) < h(e, ad) = h(b, ac) < h(a, be)$, where $h(y, zx)$ denotes the minimum distance between y and a point on the line through zx.

Take this same example, and imagine that vertices are not perturbed and thus form a regular pentagon. If we set the elevations (or data values) of a, b, c, d, e to $5, 0, 10, 0, 11$ in this sequence to assign slopes to triangles, we again have a bad example for edge-flip on the slope measure. One quick way to estimate the slope of a spatial triangle $\hat{p}_i\hat{p}_j\hat{p}_k$ given by its planar projection $p_ip_jp_k$ (as in our example) is as follows. Assume the given elevations of the vertices are such that $\pi_{k3} \geq \pi_{j3} \geq \pi_{i3}$. If $\pi_{j3} = \pi_{i3}$, then the slope is simply $\frac{\pi_{k3}-\pi_{i3}}{h(p_k, p_ip_j)}$. Otherwise, let z be the projection of a point on $\hat{p}_i\hat{p}_k$ with elevation π_{j3}. From simple calculation, z can be computed and so does the slope $\frac{\pi_{j3}-\pi_{i3}}{h(p_i, zp_j)}$.

We can stretch the same example so that it is also the Delaunay triangulation of the vertices. It thus follows that Delaunay triangulation is not necessarily the same as a min-max angle, a max-min height, or a min-max eccentricity triangulation. Trivially, we can say the same for

Delaunay triangulations and min-max slope triangulations. Since slope and eccentricity satisfy (I) but not (II), they define different optima from largest angle and height. Lastly, each pair can be distinguished by a simple example with four vertices, and thus they are all distinct.

4.1 Minimizing the Maximum Angle

A *min-max angle triangulation* of a point set S minimizes the maximum angle of its triangles, over all triangulations of S. As mentioned, such optimal triangulation has potential applications in finite element analysis and surface interpolation [BaAz76, BaLi84, Greg75].

For $x, y, z \in S$, we define $\alpha(xyz) = \max\{\angle x, \angle y, \angle z\}$ as the measure of triangle xyz. The measure of a triangulation $\mathcal{A}$ of S is $\alpha(\mathcal{A}) = \max\{\alpha(xyz) : xyz$ a triangle of $\mathcal{A}\}$. For triangle xyz, we designate vertices y with $\angle y = \alpha(xyz)$ as anchors. Then, the next lemma shows that α satisfies condition (II).

Lemma 4.2 Let xyz be a triangle of a triangulation $\mathcal{A}$ of S, and y be an anchor of xyz. Then $\alpha(\mathcal{T}) > \alpha(xyz)$ for any triangulation $\mathcal{T}$ of S that neither contains xyz nor breaks xyz at y.

Proof. Assume that xyz is not in $\mathcal{T}$ and that $\mathcal{T}$ does not break xyz at y. Then, there exists a triangle uyv in $\mathcal{T}$ so that either $u = x$ and $uv \cap yz \neq \emptyset$ (rename vertices if necessary), or uv intersects both yx and yz. In both cases, $\alpha(\mathcal{T}) \geq \angle(uyv) > \alpha(xyz)$ because $\angle xyz$ is properly contained in $\angle uyv$. $\square$

We thus establish part (1) of Theorem 4.1. We note that the edge-insertion paradigm actually evolved from the algorithm for min-max angle triangulations [ETW90], rather than the order materials are presented here. The Cake-Cutting Lemma was also the main idea behind the algorithm. Its original proof has a strong flavor of "cutting cake" from which the lemma gets its name. For this historical and sentimental reasons, we repeat the proof in the rest of this section. Though the proof may look slightly longer, it is conceptually simpler than the one in the previous chapter.

Lemma 4.3 (Cake-Cutting) Suppose triangulation $\mathcal{T}$ is an improvement of triangulation $\mathcal{A}$. Let pqr be a triangle in $\mathcal{A}$ with $\angle pqr = \alpha(\mathcal{A})$ but not in $\mathcal{T}$, and let qs be an edge in $\mathcal{T}$ that intersect pr. Let P and R be the polygons generated by adding qs to $\mathcal{A}$ and removing all edges that intersect qs. Then there are triangulations $\mathcal{P}$ and $\mathcal{R}$ of P and R with $\alpha(pqr) > \alpha(\mathcal{P})$ and $\alpha(pqr) > \alpha(\mathcal{R})$.

Proof. We prove the claim for P; it follows for R by symmetry. As before, we lay P on top of $\mathcal{T}$ so that their corresponding vertices match, and called each connected component of an edge of $\mathcal{T}$ intersected with the interior of P a *clipped edge*. Again, we use these clipped edges to cut (the cake) P to obtain $\mathcal{P}$.

Given a point x on the boundary of P, let the *path* from x to q (or x to s) be the part of the boundary between x and q (or x and s) that does not contain qs. We distinguish four classes of clipped edges xy; see Figure 4.2.

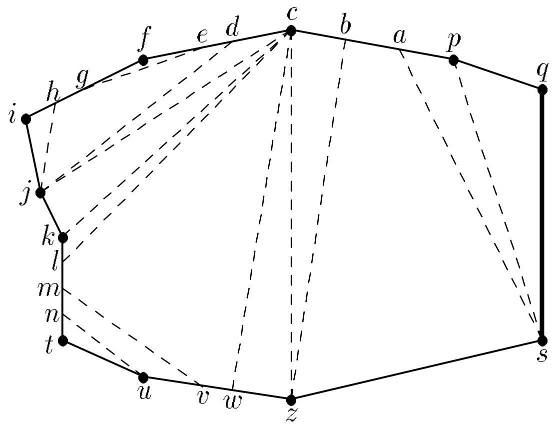

Figure 4.2: The class I edges in this example are eg and mv; the class II edges are cj, ck, cz, and sp; the class III edges are cl and cw; and the class IV edges are jh, jd, un, zb, and sa.

I. Both endpoints, x and y, are not vertices of P and thus lie on edges of P.

II. Both endpoints are vertices of P.

III. Endpoint x is a vertex of P, y is not, and y lies on the path from x to s.

IV. The same as class III except that y lies on the path from x to q.

At any vertex x of P, the clipped edges with one endpoint at x define angles at x that are all smaller than $\alpha(\mathcal{A})$, because the clipped edges come from $\mathcal{T}$. The only disadvantage of the division of P defined by the clipped edges is that some of their endpoints lie on edges of P rather than at the vertices. We will now construct a triangulation of P based on the clipped edges. It proceeds step by step where each step either removes or rotates a clipped edge or introduces a new edge.

1. All class I edges are removed. This does not harm any angle.
2. All class II edges remain where they are.
3. Let xy be a class III edge with y on the edge $\beta\gamma$ of P, where β precedes γ on the path from x to s. We replace xy by $x\gamma$.

Note first that $x\gamma$ is indeed a diagonal of P. Otherwise, it intersects the boundary of P, which implies that either x or γ is not visible from qs. This is a contradiction to the way P is constructed. Note second that the angle at x that precedes xy in the counterclockwise order increases in step 3. Still, the angle formed by $x\gamma$ is strictly contained in an angle at x in $\mathcal{A}$ because all edges of $\mathcal{A}$ that intersect the interior of P also intersect qs. It follows that the angle formed by $x\gamma$ is smaller than $\alpha(\mathcal{A})$. Another issue that comes up is that there can be class IV edges $x'y'$ with y' on the same edge $\beta\gamma$ of P; these edges now intersect $x\gamma$. To remedy this situation we replace $x'y'$ by $x'x$. By the same argument as above, $x'x$ is a diagonal of P, and the angle at x' that precedes $x'y'$ in the clockwise order, and which increases as we replace $x'y'$ by $x'x$, remains smaller than $\alpha(\mathcal{A})$.

4. If xy is a class IV edge with y on the edge $\beta\gamma$ of P, where β precedes γ on the path from x to q, then we replace xy by $x\gamma$.
5. After steps 1 through 4 we have a partial triangulation of P which we complete by adding edges arbitrarily. This finishes the construction of $\mathcal{P}$.

We have $\alpha(\mathcal{P}) < \alpha(\mathcal{A})$ since we started out with all angles smaller than $\alpha(\mathcal{A})$; each time an angle increases it remains smaller than $\alpha(\mathcal{A})$ as argued above, and step 5 decomposes angles, thus creating only smaller angles. $\square$

4.2 Maximizing the Minimum Height

Recall that the *height* $\eta(xyz)$ of triangle xyz is the minimum distance from a vertex to the opposite edge, i.e., $\eta(xyz) = \min\{h(x, yz), h(y, xz), h(z, xy)\}$. We write $\eta(\mathcal{A}) = \min\{\eta(xyz) : xyz \text{ a triangle of } \mathcal{A}\}$ for the measure of a triangulation $\mathcal{A}$ of S. A *max-min height triangulation* of S maximizes $\eta(\mathcal{A})$ over all triangulations $\mathcal{A}$ of S. Such triangulations have been suggested for use in surface interpolation [GCR77, WaPh84].

We next show that $-\eta$ satisfies condition (II), when we define vertices of xyz with largest angle to be anchors. This establishes part (2) of Theorem 4.1.

Lemma 4.4 Let xyz be a triangle of a triangulation $\mathcal{A}$ of S, and let y be an anchor of xyz. Then $\eta(\mathcal{T}) < \eta(xyz)$ for any triangulation $\mathcal{T}$ of S that neither contains xyz nor breaks xyz at y.

Proof. Since y is an anchor, we have height $\eta(xyz) = h(y, zx)$, which is the distance between y and a point $s \in zx$. Assume that xyz is not in $\mathcal{T}$ and that $\mathcal{T}$ does not break xyz at y. Then there exists a triangle uyv in $\mathcal{T}$ so that either $u = x$ and $uv \cap yz \neq \emptyset$ (rename vertices if necessary), or uv intersects both yx and yz. In both cases, $\eta(uyv) \leq h(y, uv) < \eta(xyz)$ because $uv \cap ys \neq \emptyset$. $\square$

We insert the following paragraph on the relations of non-obtuse triangulations with some of the above optimal triangulations. These, as we will see, have some implication to the implementation of the edge-insertion algorithm. We call a triangulation *non-obtuse* if the sizes of all angles, three per triangle, are less than or equal to $\frac{\pi}{2}$. It should be clear that not every point set S admits a non-obtuse triangulation. If $\mathcal{T}$ is a non-obtuse triangulation of S, then it is the Delaunay triangulation, or a completion of it in degenerate cases. This is because $\mathcal{T}$ is optimal with respect to the edge-flip operation. Besides those properties of Delaunay triangulations (Section 2.4), a non-obtuse triangulation $\mathcal{T}$ of a point set also minimizes the maximum angle since it is unique up to choosing diagonals of rectangles. The next lemma implies that $\mathcal{T}$ also maximizes the minimum height.

Lemma 4.5 If $\mathcal{T}$ is a non-obtuse triangulation of S, then there is no other triangulation that improves $\mathcal{T}$ with respect to η.

Proof. We show that there is no edge whose insertion could improve $\mathcal{T}$; the Cake-Cutting Lemma then implies the assertion. Let pqr be a triangle in $\mathcal{T}$ with $\eta(pqr) = h(q, rp) = \eta(\mathcal{T})$, and assume there is an edge qs, with $qs \cap rp \neq \emptyset$, whose insertion improves $\mathcal{T}$. Let pqt be the triangle in the improvement that lies on the same side of the line through pq as pqr. If $h(p, sq) \leq h(q, rp)$, then $\eta(pqt) \leq h(p, tq) \leq h(p, sq) \leq h(q, rp) = \eta(pqr)$, a contradiction to the assumption that pqt is a triangle in the improvement of $\mathcal{T}$ but not in $\mathcal{T}$. Therefore, $h(p, sq) > h(q, rp)$, and symmetrically $h(r, sq) > h(q, rp)$. It follows that $\angle pqs > \angle qpr$ and $\angle rqs > \angle qrp$. But now $\angle pqs + \angle rqs = \angle pqr > \angle qpr + \angle qrp$ which can be the case only if $\angle pqr > \frac{\pi}{2}$, a contradiction to the assumption. $\square$

Lemma 4.5 implies that during the construction of a max-min height triangulation the heights of non-obtuse triangles need not be stored in the priority queue. This is because the height of such a triangle can be smallest only if the current triangulation is already optimal.

4.3 Minimizing the Maximum Slope

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$ defining a surface $x_3 = f(x_1, x_2)$ in $\mathbb{R}^3$. The *gradient* of f is the vector $\nabla f = (\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2})$, each component of which is itself a function from $\mathbb{R}^2$ to $\mathbb{R}$. Define $\nabla^2 f = (\frac{\partial f}{\partial x_1})^2 + (\frac{\partial f}{\partial x_2})^2$, and call $\sqrt{\nabla^2 f}$ at a point (x_1, x_2) the *slope* at this point.

Let S be a set of n points in $\mathbb{R}^2$ and let $\hat{S}$ be the corresponding set in $\mathbb{R}^3$ where each point of S has a third coordinate called *elevation*. For a point x of S, we write $\hat{x}$ for the "lifted" point, that is, the corresponding point in $\hat{S}$. Analogous to the definitions in $\mathbb{R}^2$, $\hat{x}\hat{y}$ denotes the line segment with endpoints $\hat{x}$ and $\hat{y}$, and $\hat{x}\hat{y}\hat{z}$ denotes the triangle with vertices $\hat{x}, \hat{y}, \hat{z}$. We can think of $\hat{x}\hat{y}\hat{z}$ as a partial function f on $\mathbb{R}^2$, defined within xyz. At each point in the interior of xyz, the gradient is well defined and the same as for any other point in the interior of xyz. We can therefore set $\sigma(xyz)$ equal to the slope at any point in the interior of xyz, and call it the *slope* of xyz. For a triangulation $\mathcal{A}$ of S define $\sigma(\mathcal{A}) = \max\{\sigma(xyz) : xyz$ a triangle of $\mathcal{A}\}$, as usual. A *min-max slope triangulation* of S minimizes $\sigma(\mathcal{A})$ over all triangulations $\mathcal{A}$ of S.

Observe that the direction of steepest descent at any point in the interior of a triangle xyz is given by $\Delta = -\nabla f$ at that point. We call vertex y an *anchor* of xyz unless the line $y + \lambda\Delta$, $\lambda \in \mathbb{R}$, intersects the closure of xyz only at y. In the non-degenerate case, xyz has only one anchor, but if Δ is parallel to an edge, then there are two anchors. Call the intersection of the closure of $\hat{x}\hat{y}\hat{z}$ with the plane parallel to the x_3-axis through $y + \lambda\Delta$ the *descent line* $\ell(xyz)$ of xyz, assuming y is an anchor of xyz.

The remainder of this section shows that measure σ satisfies condition (I). For technical reasons it is necessary to assume that no four points of $\hat{S}$ are coplanar. Indeed, the strict inequality in the next lemma is incorrect without this assumption. This general position assumption, however, does not diminish the generality of our algorithm, because a simulated perturbation of the points can be used to enforce general position [EdMü90]. This perturbation is infinitesimal. Consider the triangulation of the unperturbed points that corresponds to an optimal triangulation of the perturbed points. This triangulation must minimize the maximum slope over all triangulations of the unperturbed points.

Lemma 4.6 Let xyz be a triangle of a triangulation $\mathcal{A}$ of S, and let y be an anchor of xyz. Then $\max\{\sigma(\mathcal{A}), \sigma(\mathcal{T})\} > \sigma(xyz)$ for every triangulation $\mathcal{T}$ of S that neither contains xyz nor breaks xyz at y.

Proof. The slope of xyz, $\sigma(xyz)$, is also the slope of the descent line $\ell_1 = \ell(xyz)$. Assume without loss of generality that ℓ_1 descends from $\hat{y}$ down to where it meets the closure of $\hat{x}\hat{z}$. (If it ascends, we use the same argument only with the x_3-axis reversed.) Assume also that $\mathcal{T}$ neither contains xyz nor breaks it at y. It follows that $\mathcal{T}$ contains an edge uv so that either $u = x$ and $uv \cap yz \neq \emptyset$ (rename vertices if necessary), or uv intersects both yx and yz. If $\sigma(uyv) > \sigma(xyz)$, then $\sigma(\mathcal{T}) > \sigma(xyz)$ and there is nothing to prove.

Otherwise, the edge $\hat{u}\hat{v}$ must pass *above* ℓ_1 in $\mathbb{R}^3$. By this we mean that there is a line parallel to the x_3-axis that meets $\hat{u}\hat{v}$ and ℓ_1, and the elevation of its intersection with $\hat{u}\hat{v}$ exceeds the elevation of its intersection with ℓ_1, as in Figure 4.3. Then at least one of $\hat{u}$ and $\hat{v}$ must lie

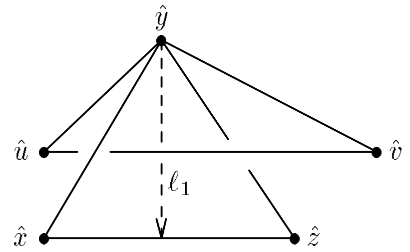

Figure 4.3: The triangle xyz with anchor y in $\mathcal{A}$ is neither contained in $\mathcal{T}$ nor is it broken at y by $\mathcal{T}$. Therefore, $\mathcal{T}$ contains a triangle uyv that intersects xyz as shown. It is possible that $u = x$ or $v = z$, but not both at the same time.

above the plane h_1 through points $\hat{x}, \hat{y}, \hat{z}$; say $\hat{v}$ lies above h_1. Consider the triangle yvz, and note that it is not necessarily a triangle of $\mathcal{A}$ or $\mathcal{T}$, nor even an empty triangle of S. We have $\sigma(yvz) > \sigma(xyz)$ because the projection (along x_3-axis) of ℓ_1 onto the plane h_2 through $\hat{y}, \hat{v}, \hat{z}$ is steeper than ℓ_1 but not steeper than $\ell_2 = \ell(yvz)$. We distinguish three cases depending on which vertex is the anchor of yvz, that is, through which one a line of steepest descent of $\hat{y}\hat{v}\hat{z}$ passes.

Case 1. v is an anchor of yvz. Then ℓ_2 connects $\hat{v}$ with a point on the closure of $\hat{y}\hat{z}$. Consider the intersection of $\mathcal{A}$ with a plane parallel to the x_3-axis through ℓ_2. This intersection includes a polygonal chain that connects $\hat{v}$ with that same point on the closure of $\hat{y}\hat{z}$ (since yz is an edge in $\mathcal{A}$). One of the segments in the chain must have slope at least the average slope of the chain; hence one of the triangles abc in $\mathcal{A}$ has $\sigma(abc) \geq \sigma(yvz) > \sigma(xyz)$, and $\sigma(\mathcal{A}) > \sigma(xyz)$.

Case 2. z is an anchor of yvz. Then ℓ_2 connects $\hat{z}$ with a point on the closure of $\hat{y}\hat{v}$. Then we use the same argument as in Case 1, only applied to $\mathcal{T}$. Since yv is an edge in $\mathcal{T}$ at least one of the triangles abc in $\mathcal{T}$ that intersect the projection (along x_3-axis) of ℓ_2 has $\sigma(abc) \geq \sigma(yvz) > \sigma(xyz)$, and therefore $\sigma(\mathcal{T}) > \sigma(xyz)$.

Case 3. y is an anchor of yvz. In this case ℓ_2 connects $\hat{y}$ with a point $\hat{w}$ on the closure of $\hat{v}\hat{z}$. Furthermore, it is impossible that ℓ_2 descends from $\hat{y}$ to $\hat{w}$ because $\hat{w}$ lies above h_1, which contradicts $\sigma(yvz) > \sigma(xyz)$. Thus, it must be that ℓ_2 descends from $\hat{w}$ down to $\hat{y}$. Then $\sigma(uyv) > \sigma(yvz)$ because $\hat{u}\hat{v}$ passes above ℓ_2. But $\sigma(yvz) > \sigma(xyz)$, so we have shown $\sigma(\mathcal{T}) > \sigma(xyz)$. $\square$

This implies part (3) of Theorem 4.1. An example to show that σ indeed violates (II) is given in Figure 4.4, so an $O(n^2 \log n)$ time algorithm for min-max slope triangulations seems out of reach at this moment.

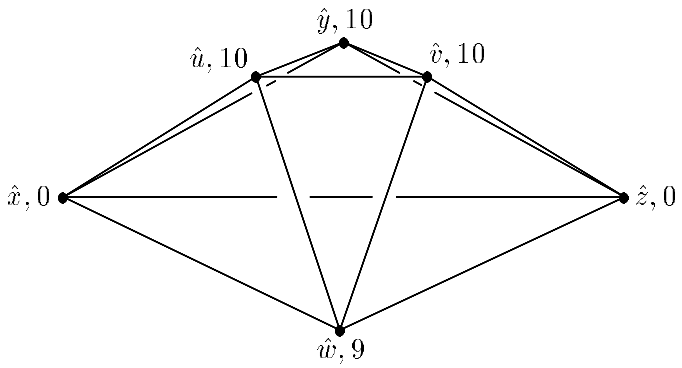

Figure 4.4: Triangulation $\mathcal{T}$ with diagonals uv, vw, and wu is an improvement of $\mathcal{A}$ with diagonals xy, yz, and zx. $\mathcal{T}$ has no triangle with slope as large as $\sigma(xyz)$, but does not break xyz at any of its vertices.

4.4 Minimizing the Maximum Eccentricity

Consider a triangle xyz and let (c_1, ρ_1) be its circumcircle, with center c_1 and radius ρ_1. Recall that the *eccentricity* of xyz, $\epsilon(xyz)$, is the infimum over all distances between c_1 and points in the closure of xyz. Clearly, $\epsilon(xyz) = 0$ iff c_1 lies in the closure of xyz. Note that eccentricity is related to the size of the largest angle, $\alpha(xyz)$. Specifically, unless $\epsilon(xyz) = \epsilon(abc) = 0$,

$$\alpha(xyz) < \alpha(abc) \quad \text{iff} \quad \frac{\epsilon(xyz)}{\rho_1} < \frac{\epsilon(abc)}{\rho_2}$$

where ρ_2 is the radius of the circumcircle of abc. This suggests we call y an *anchor* of xyz if the angle at y is at least as large as the angles at x and z. As usual, we define $\epsilon(\mathcal{A}) = \max\{\epsilon(xyz) : xyz \text{ a triangle of } \mathcal{A}\}$. A *min-max eccentricity triangulation* of S minimizes $\epsilon(\mathcal{A})$ over all triangulations $\mathcal{A}$ of S (see [WaPh84]). Obviously, a non-obtuse triangulation, when it exists, for a point set S also minimizes its maximum eccentricity.

Figure 4.5 shows that measure ϵ does not satisfy condition (II). On the other hand, ϵ does satisfy condition (I) as shown by the next lemma, which then implies part (4) of Theorem 4.1.

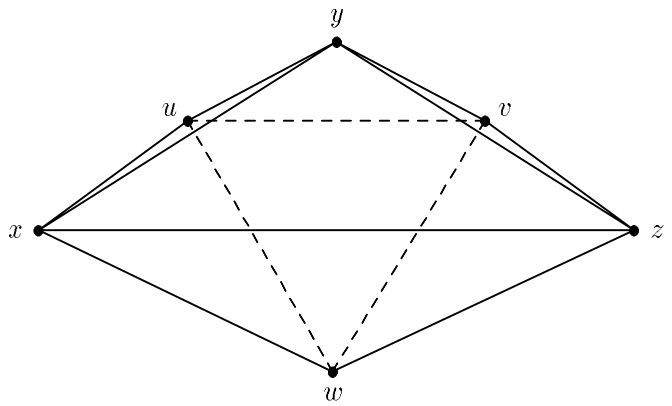

Figure 4.5: $\mathcal{T}$ is the triangulation with diagonals uv, vw, and wu, and $\mathcal{A}$ is the one with diagonals xy, yz, and zx. We have $\epsilon(\mathcal{T}) = \epsilon(uyv)$. As vertices u and v lie very close to yx and yz, respectively, the circumcircle of uyv is significantly smaller than the one of xyz and $\epsilon(uyv) < \epsilon(xyz)$. So, $\mathcal{T}$ is an improvement of $\mathcal{A}$, yet $\mathcal{T}$ does not break xyz of $\mathcal{A}$ at any of its vertices.

Lemma 4.7 Let xyz be a triangle of a triangulation $\mathcal{A}$ of S, such that $\epsilon(xyz) > 0$, and let y be an anchor of xyz. Then $\max\{\epsilon(\mathcal{A}), \epsilon(\mathcal{T})\} > \epsilon(xyz)$ for every triangulation $\mathcal{T}$ of S that neither contains xyz nor breaks xyz at y.

Proof. Assume that $\mathcal{T}$ neither contains xyz nor breaks it at y. Therefore, $\mathcal{T}$ contains a triangle uyv so that $u = x$ and $uv \cap yz \neq \emptyset$ (rename vertices if necessary), or uv intersects yx and yz. Let (c_1, ρ_1) be the circumcircle of xyz. If neither u nor v are enclosed by this circle, then $\epsilon(\mathcal{T}) \geq \epsilon(uyv) > \epsilon(xyz)$. Otherwise, assume that v is enclosed by (c_1, ρ_1) and consider the line segment c_1v. It intersects a sequence of edges of $\mathcal{A}$, ordered from c_1 to v. For an edge ab in this sequence let abc be the supporting triangle so that c and c_1 lie on different sides of the line through ab. Assume that ab is the first edge in the sequence so that (c_1, ρ_1) encloses c but not a and not b. Then $\epsilon(\mathcal{A}) \geq \epsilon(abc) > \epsilon(xyz)$. □

4.5 Discussion

This chapter shows that the edge-insertion paradigm can optimize the min-max angle, the max-min height, the min-max slope and the min-max eccentricity criteria. These polynomial time solutions are the first, and currently the only ones, for these problems. An interesting side result is that non-obtuse triangulations also optimize the above criteria except for the min-max slope.

We remark that conditions (I) and (II) have also been tested for other measures mentioned earlier in Section 1.3. None of them seems to meet the requirements. Of course, the techniques of retriangulating regions and searching for a good edge-insertion can still serve as a heuristic that is more powerful than edge-flip when applied to these problems.

Acknowledgments

The results collected in the last two chapters come from several people. It all started as a course project, suggested by Professor Herbert Edelsbrunner in January 1989, studying the problem of min-max angle triangulations. Harald Rosenberger and Roman Waupotitsch first observed that four vertices are enough to distinguish min-max angle from the Delaunay triangulation. This injected some excitement to the problem. Subsequently, I took up the project and worked under the guidance of Herbert Edelsbrunner to solve the problem in time $O(n^8)$ with the basic version of the edge-insertion paradigm. Later we improved it to time $O(n^3)$ and storage $O(n^2)$ with the ear-cutting process as in Section 3.4.1. Then, Roman Waupotitsch observed that the search for a proper edge can be done efficiently by stepping from triangle to triangle, which improves the storage to linear because there is no need for a bit array that keeps track of the quadratically many possible edges. This effort leads to our subsequent discovery of the $O(n^2 \log n)$ time and linear storage algorithm for the problem. This is reported in [ETW90].

The thought of formulating the solution to min-max angle triangulation into a general method was considered for some time. In December 1990, I observed that the method, with the inductive rather than the constructive proof of the Cake-Cutting Lemma, also solves the max-min height criterion in the same amount of time and storage. From there, condition (II), similar to the present form, was formulated. In June 1991, I learned from Marshall Bern (Xerox PARC) that Scott Mitchell (Cornell University) has found a similar application of the edge-insertion paradigm to maximizing triangle heights. When Herbert Edelsbrunner visited Xerox

PARC in August 1991, he solved the min-max slope problem with Marshall Bern and David Eppstein (University of California, Irvine), by relaxing condition (II) to (I). Subsequently, Marshall Bern found the solution to the min-max eccentricity problem. These results are collected in [BEEMT92]. Roman Waupotitsch has implemented and experimented with most of these algorithms [EdWa92, Waup92]. The programs are currently available via anonymous ftp from the directory "/SGI/MinMaxer" at site "ftp.ncsa.uiuc.edu".

Chapter 5

Minimizing the Maximum Length

This chapter is devoted to the study of the min-max length criterion. Generally speaking, length criteria are known to be difficult to optimize; no optimal triangulations defined on (non-trivial) length criteria has previously been computed efficiently. In the face of this apparent difficulty, the quadratic time algorithm on constructing min-max length triangulations for point sets presented here is somewhat surprising. There is evidence for the potential usefulness of such a triangulation [BrZl70, WGS90].

The developments in the forthcoming sections to solve the min-max length problem are elementary, but lengthy and occasionally involved. Some of the details contribute insight into edge length criteria. Before we continue, let us first rule out some seemingly promising approaches to computing min-max length triangulation. Note first that the Delaunay triangulation does not minimize the maximum edge length, as shown in Figure 5.1. Second, the iterative methods that use the edge-flip (Section 2.3) or the more general edge-insertion operation (Chapter 3) can get caught in local optima; see Figure 5.2.

Third, the incremental greedy method, that repeatedly adds the shortest edge that does not intersect any previously added edges (see, for example, [LeLi90]), also fails to minimize the maximum edge length. For the six points in Figure 5.2, this method computes the triangulation with diagonals xy, wu, and uy. Finally, let us take a brief look at the decremental greedy method that throws away edges in the order of decreasing length. It stops the deletion process if another

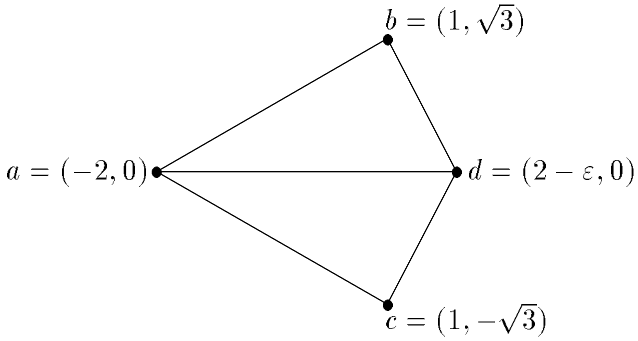

Figure 5.1: The length of the longest edge in the shown Delaunay triangulation approaches $\frac{2}{\sqrt{3}}$ the length of the longest edge in the other triangulation, as ε approaches 0. Indeed, $\frac{2}{\sqrt{3}}$ is the worst possible ratio achieved by a Delaunay triangulation. This follows from the fact that it minimizes the radius, r, of the maximum smallest enclosing circle of all triangles [Raja91]. Hence no edge in the Delaunay triangulation exceeds $2r$ and every other triangulation has an edge of length at least $r\sqrt{3}$.

deletion would render the set of edges so that it does not contain any triangulating subset. The trouble with this approach is that it is not clear how to efficiently decide whether the evolving edge set is still sufficient to triangulate the point set. Indeed, Lloyd [Lloy77] proves that the general version of this problem (decide whether a given edge set contains a triangulation) is NP-complete.

The organization of this chapter is as follows. Section 5.1 formulates the global algorithm; its straightforward implementation using dynamic programming takes time $O(n^3)$. The only

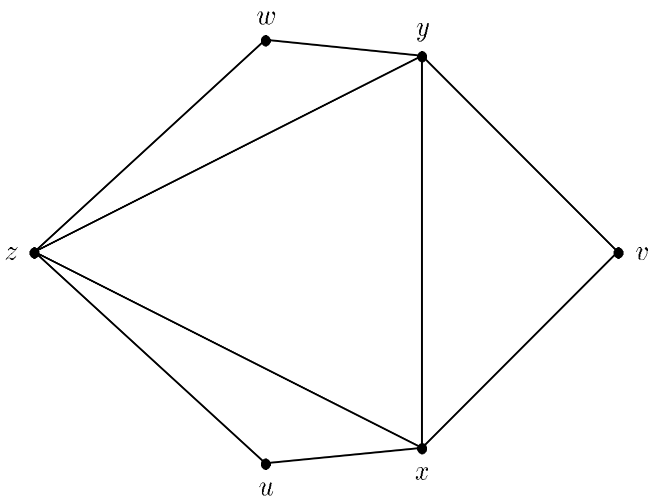

Figure 5.2: The diagonals are such that $|xy| < |uv| = |vw| < |wu| < |yz| = |zx| < |uy| = |xw| < |zv|$. So, neither edge-flip nor edge-insertion can locally improve the shown triangulation $\mathcal{A}$ containing diagonals xy, yz and zx. However, the min-max length triangulation is not $\mathcal{A}$ but the one with diagonals uv, vw, and wu.

intricate part of this algorithm is the proof of correctness provided in Section 5.2. Sections 5.3 to 5.6 present a specialized polygon triangulation algorithm that can be used to speed up the general algorithm to time $O(n^2)$; Section 5.7 discusses implementation issues. While Sections 5.1 through 5.6 assume that the Euclidean metric is used to measure length, Section 5.8 demonstrates that all results extend to general normed metrics. Indeed, the arguments in Sections 5.1 through 5.6 are axiomatically derived from a few basic lemmas in order to minimize the number of changes necessary to generalize the results. Finally, Section 5.9 summarizes the contribution of this chapter.

5.1 The Global Algorithm

Let S be a point set. We define $ch(S)$ as the smallest convex polygon whose closure contains the convex hull of S. In the case where three or more collinear points of S lie on the boundary of this polygon, we think of each such points as a vertex of the polygon. Thus, edges are taken only between adjacent collinear points. Each edge of $ch(S)$ is an edge of *every* triangulation of S, and therefore also of every min-max length triangulation.

The circle with center x and radius ρ is denoted by (x, ρ). Recall (from Section 2.4) that $rng(S)$ stands for the relative neighborhood graph of S, and the lune of ab is the intersection of the two disks bounded by $(a, |ab|)$ and $(b, |ab|)$. Edge set of $rng(S)$ contains ab iff $|ab| \leq \min_{x \in S-\{a,b\}} \max\{|xa|, |xb|\}$, i.e., the lune of ab is disjoint from S. We will see in the next section that there is a min-max length triangulation $mlt(S)$ that contains all edges of $rng(S)$. For convenient, $ch(S)$ and $rng(S)$ will be interpreted as edge sets where appropriate.

Because $ch(S) \cup rng(S)$ is a connected graph, it decomposes the convex hull of S into simply connected regions. Though the boundary of each region is not necessarily a simple polygon, we can always treat it as one computationally (Section 1.2). Thus, we can construct $mlt(S)$ by computing $ch(S) \cup rng(S)$ and then (optimally) triangulating each resulting simple polygon.

Input. A set S of n points in $\mathbb{R}^2$.

Output. A min-max length triangulation of S.

Algorithm. 1. Construct $ch(S)$ and $rng(S)$.
2. Determine the polygons defined by $ch(S) \cup rng(S)$.
3. Find a min-max length triangulation for each such polygon.

Step 1 can be carried out in time $O(n \log n)$ using results documented in [PrSh85, Supo83]. Using the quad-edge data structure of Guibas and Stolfi (Section 1.2) for storing the plane geometric graph $ch(S) \cup rng(S)$, step 2 can be accomplished in time $O(n)$. Then, at step 3, we can use dynamic programming (Section 2.2) to compute an optimal triangulation for each polygon in time cubic and storage quadratic in the number of its vertices. This adds up to time $O(n^3)$ and storage $O(n^2)$. The correctness of the algorithm is established next.

5.2 The Subgraph Theorem

The main result of this section is what we call the Subgraph Theorem mentioned earlier. We begin with two elementary lemmas about distances between four points in convex and in non-convex position.

$\square$-Lemma. For a convex quadrilateral $abcd$, we have $|ab| + |cd| < |ac| + |bd|$.

Proof. Let x be the intersection point of the two diagonals, ac and bd. Clearly, $|ab| + |cd| < (|ax| + |xb|) + (|cx| + |xd|) = |ac| + |bd|$. $\square$

In words, the total length of the two diagonals of a convex quadrilateral always exceeds the total length of two opposite edges. This is true even if three of the four vertices are collinear. It implies that if one diagonal is no longer than one of the edges, then the other diagonal is longer than the opposite edge.

Δ-Lemma. Let a, b, c, d be four distinct points so that the closure of the triangle abc contains d. Then $|ad| < \max\{|ab|, |ac|\}$.

Proof. If a, b, c, d are collinear the result is obvious. Otherwise, let d' be the intersection of the edge bc with the line through ad, and note that $|ad| \leq |ad'|$. Of all points on $bc \cup \{b, c\}$, only the endpoints (b and c) can possibly maximize the distance to a. The assertion follows because if d' is an endpoint of bc, then $d \neq d'$ and therefore ad is strictly shorter than ad'. $\square$

Note that the length of the longest edge of any minimum spanning tree is no longer than the longest edge of any triangulation of S. This follows trivially from the fact that every triangulation contains a spanning tree. It is not very difficult to prove that the same is true for the relative neighborhood graph of S. Note that the *bisector* of two points p and q is the set of points equidistant to both.

Length Lemma. Every triangulation of S contains an edge that is at least as long as the longest edge of $rng(S)$.

Proof. Let pq be the longest edge of $rng(S)$ and let $\mathcal{A}$ be an arbitrary triangulation of S. If $pq \in \mathcal{A}$, there is nothing to prove. Otherwise, pq intersects edges $r_1s_1, r_2s_2, \ldots, r_ks_k$ of $\mathcal{A}$, sorted from p to q, with all r_i on one side of the line through pq and all s_i on the other. If pq is longer than all edges in $\mathcal{A}$, then r_1 and s_1 are both inside the circle $C_p = (p, |pq|)$, because pr_1 and ps_1 are both edges of $\mathcal{A}$. By the definition of $rng(S)$, r_1 and s_1 are thus outside or on the circle $C_q = (q, |pq|)$. Therefore, r_1 and s_1 lie in the half-plane of points closer to p than to q. Symmetrically, r_k and s_k lie inside C_q and outside or on C_p and therefore in the half-plane of points closer to q than to p. For each $1 \leq i \leq k-1$, we have either $r_i = r_{i+1}$ or $s_i = s_{i+1}$, which implies that there is an index j so that r_j and s_j do not lie on the same side of the bisector of pq. But then the $\square$-Lemma implies that $|r_js_j| > |pq|$, because $|pq|$ is no longer than each of two opposite edges of the convex quadrilateral pr_jqs_j, a contradiction. $\square$

The proof of the Subgraph Theorem is similar to that of the Length Lemma, although considerably more involved. The basic idea is to assume an extremal counterexample and to contradict its existence by retriangulating parts of it using no long edges. In the following, we first develop three facts showing the possibilities of retriangulations, and then prove the theorem.

Let $\mathcal{T}$ be a min-max length triangulation of S that does not contain some edge pq of $rng(S)$. Suppose pq intersects the triangles $t_1, t_2, \ldots, t_k$ of $\mathcal{T}$, sorted from p to q (see Figure 5.3 left). The deletion of the edges that intersect pq would result in a simply connected region, whose boundary can be interpreted as a simple polygon—we treat each edge in its boundary as a pair of edges with opposite directions, and to trace the boundary of the region we traverse all directed edges that have the region on their left side. Any two consecutive (directed) edges

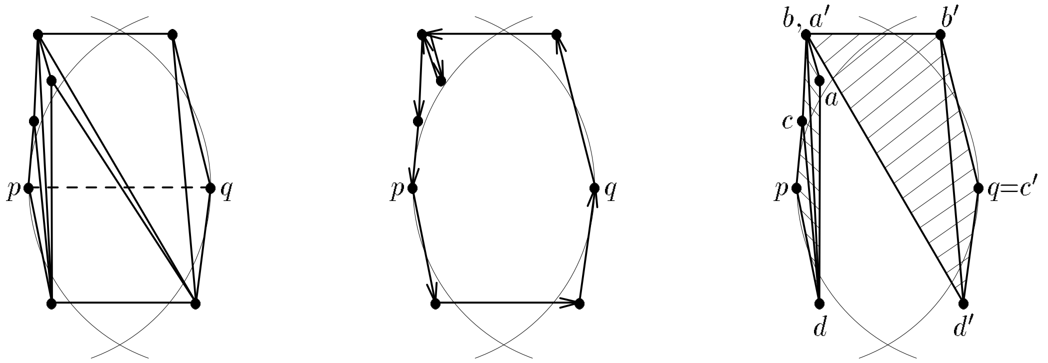

Figure 5.3: To the left we see the triangles of $\mathcal{T}$ that intersect pq. If we remove the edges intersecting pq we get a polygon whose boundary is oriented in a counterclockwise order. The prefix P and the suffix Q defined for this configuration are illustrated to the right. Although b and a' are the same point, they refer to different angles of this point.

define an *angle* (see Figure 5.3 middle). Note that a vertex can correspond to many angles, although the common situation is that it corresponds only to one. We will therefore sometimes ignore the difference between vertices and corresponding angles. Points p and q correspond to only one angle each. An angle is *convex* if the two defining edges form a left turn. Call the sequence of edges from p to q the *lower chain* and the sequence from q to p the *upper chain*. Each chain contains at least one convex angle different from p and q.

A *prefix* is an initial subsequence of $t_1, t_2, \ldots, t_k$, and a *suffix* is a terminal subsequence of $t_1, t_2, \ldots, t_k$. We say that a prefix (suffix) *covers* an angle of the polygon if it contains all triangles incident to this angle. Let i be minimal so that the prefix $P = t_1, t_2, \ldots, t_i$ covers a convex angle other than p, and let j be maximal so that the suffix $Q = t_j, t_{j+1}, \ldots, t_k$ covers a convex angle other than q. P and Q consist of at least two triangles each. We let b be the convex angle (vertex) covered by P—it is incident to both t_i and t_{i-1}—and d be the other vertex common to t_i and t_{i-1}. Furthermore, c is the third vertex of t_{i-1} and a is the third vertex of t_i (see Figure 5.3 right). Symmetrically, define vertices b', d', c', a' of Q. We say that P (Q) is *type* 1 if the last (first) two triangles of P (Q) are the only ones incident to b (b'), and it is *type* 2, otherwise (see Figure 5.4). If P is type 1, then a, b, c belong to the same chain and d belongs to the other chain (this includes the case that $c = p$), and if P is type 2, then a, b belong to one chain and c, d to the other.

Fact 5.1 $P = t_1, t_2, \ldots, t_i$ and $Q = t_j, t_{j+1}, \ldots, t_k$ share at most two triangles, that is, $i-1 \leq j$.

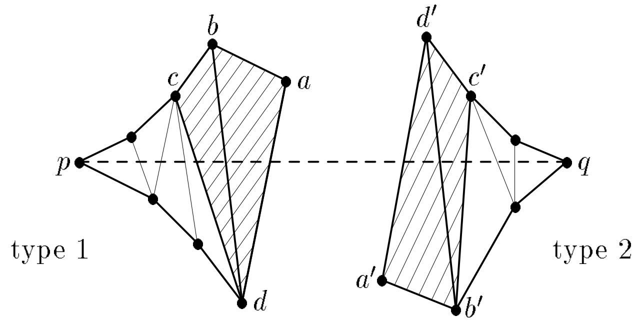

Figure 5.4: The prefix P with vertices a, b, c, d and the suffix Q with vertices a', b', c', d' are defined depending on pq. P is type 1 and Q is type 2. For illustration purposes the constraint that all vertices must lie outside the lune of pq has been ignored.

Proof. We show that the suffix $R = t_{i-1}, t_i, \ldots, t_k$ covers at least one convex angle other than q, so Q cannot be bigger than R. If P is type 1, then R covers b, which is convex. Otherwise, R covers all angles between d and q, d included. Since all angles between p and d, p and d excluded, are non-convex, at least one angle between d and q must be convex, and this angle is covered by R. $\square$

It should be clear that $abcd$ and $a'b'c'd'$ are both convex quadrilaterals by the choice of their vertices. The next two facts imply that either $abcd$, or $a'b'c'd'$, or both have alternate triangulations using ac or $a'c'$, while maintaining the maximum edge length of $\mathcal{T}$. In other words, bd, or $b'd'$, or both can be switched. Formally, we call bd ($b'd'$) *switchable* if ac ($a'c'$) is no longer than the longest edge of $\mathcal{T}$. Fact 5.2 shows strong locality constraints for a and d (a' and d') if bd ($b'd'$) is not switchable. Define

$$A = \{x \in \mathbb{R}^2 : |xp| > |pq| \text{ and } |xp| > |xq|\} \text{ and}$$

$$D = \{x \in \mathbb{R}^2 : |xp| \geq |pq| \text{ and } |xq| < |pq|\},$$

with the understanding that A and a belong to one half-plane defined by the line through pq, and D and d belong to the other (see Figure 5.5).

Fact 5.2 If bd is not switchable, then $a \in A$ and $d \in D$.

Proof. Since bd is not switchable, ac must be longer than the other five edges defined by a, b, c, d, and, by the Length Lemma, it must be longer than pq. We first show that $|ac| \leq |ap|$ and then derive the four inequalities needed to establish the claim.

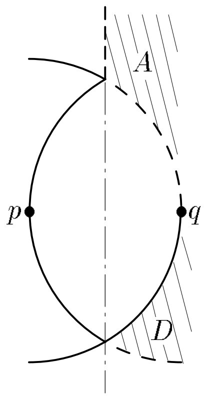

Figure 5.5: The regions A and D as defined for the case when a is on the upper chain.

(i) $|ac| \leq |ap|$. We can assume that $c \neq p$. Note that c is contained in the closure of bdp. Since the line through bd separates a from p, the closures of abp and adp cover bdp completely, and therefore one of them contains c. If c lies in the closure of abp, the claim follows from $|ab| < |ac|$ and the Δ-Lemma for abp, and if c lies in adp, it follows from $|ad| < |ac|$ and the Δ-Lemma for adp.

(ii) $|ap| > |pq|$. This follows from (i) and $|ac| > |pq|$ implied by the Length Lemma.

(iii) $|dq| < |pq|$. Assume $|dq| \geq |pq|$. The $\square$-Lemma for $paqd$ implies $|ad| > |ap|$, and thus $|ad| > |ac|$ because of (i), a contradiction.

(iv) $|dp| \geq |pq|$. This is immediate from (iii) because pq is an edge of $rng(S)$.

(v) $|ap| > |aq|$. Assume $|ap| \leq |aq|$. By the $\square$-Lemma for $paqd$ and (iv), we get $|ad| > |aq|$. This implies $|ad| > |ap|$ by assumption, and $|ad| > |ac|$ by (i), a contradiction.

The proof of Fact 5.2 is now complete because (ii) and (v) are equivalent to $a \in A$ and (iii) and (iv) are equivalent to $d \in D$. $\square$

Symmetrically, we define regions A' and D' which is where a' and d' must lie if $b'd'$ is not switchable. Using Facts 5.1 and 5.2 we can now show that there is always an edge that can be switched.

Fact 5.3 It is not possible that both bd and $b'd'$ are non-switchable.

Proof. If bd and $b'd'$ are both non-switchable, then ad lies on q's side of the bisector of pq and $a'd'$ lies on p's side, by Fact 5.2. Because of Fact 5.1 and because ad is the last edge of P and $a'd'$ is the first edge of Q, we have $\{a, d, a', d'\} = \{a, b, c, d\} = \{a', b', c', d'\}$. Furthermore, the fact that bd and $b'd'$ are both edges of $\mathcal{T}$ implies that they are the same and thus $b = d'$, $d = b'$, $a = c'$, $c = a'$ (see Figure 5.6). It follows that the polygon has the shape of a diamond

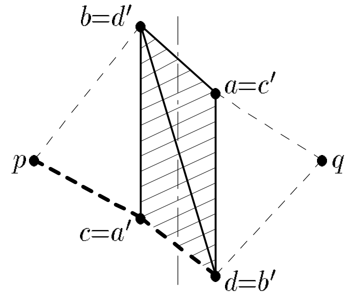

Figure 5.6: If bd and $b'd'$ are both non-switchable, then b and d are the only convex angles besides p and q.

with p, b, q, d as the only convex angles. This contradicts the locality constraints for a, b, c, d stated in Fact 5.2. In particular, the chain from p to $d \in D$ is concave or straight and therefore enclosed by the circle $(q, |pq|)$. It follows that this chain is disjoint from A', which is where $c = a'$, the predecessor of d in this chain, is supposed to lie. ◻

With the above results and notations, we now choose an extremal counterexample to prove the main result of this section.

Subgraph Theorem. Every point set S in $\mathbb{R}^2$ has a min-max length triangulation $mlt(S)$ so that $rng(S) \subseteq mlt(S)$.

Proof. We assume there is a set S so that no min-max length triangulation contains $rng(S)$. Let $\mathcal{T}$ be a min-max length triangulation of S that satisfies the following extremal properties, where later properties are contingent upon earlier ones.

(1) $\mathcal{T}$ minimizes the number of edges that intersect pq.

(2) $\mathcal{T}$ minimizes the number of edges incident to b that intersect pq.

(3) $\mathcal{T}$ minimizes the number of edges incident to b' that intersect pq.

It is conceivable that $\mathcal{T}$ is not unique, but it will be sufficient to assume that $\mathcal{T}$ is any one of the remaining triangulations.

By Fact 5.3, either bd, or $b'd'$, or both are switchable. If bd is switchable and P is type 1, then the number of edges that intersect pq decreases when bd is switched. This contradicts property (1). Thus, P must be type 2 if bd is switchable, and, similarly, Q must be type 2 if $b'd'$ is switchable. When we switch bd the degree of b decreases, which contradicts property (2). Thus, it must be that bd is not switchable and $b'd'$ is. But switching $b'd'$ decreases the degree of b', which would contradict property (3), unless the degree of b increases at the same time. Remember that (3) is contingent upon (2), so if (2) is not satisfied any more, then we cannot draw any conclusion. Thus, the configuration left for analysis is as shown in Figure 5.7.

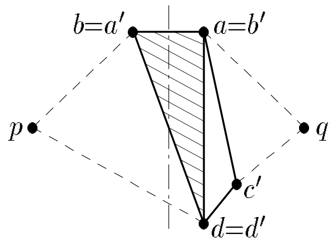

Figure 5.7: In the final configuration bd is non-switchable, so $a \in A$ and $d \in D$, and $b'd'$ is switchable, so Q is type 2. Furthermore, switching $b'd'$ to $a'c'$ increases the degree of b, so $a' = b$ and therefore P and Q overlap in exactly one triangle. The figure ignores that by rights all points should lie outside the lune of pq.

To reach the final contradiction, we switch $b'd'$ and redefine Q based on the new configuration. Since all angles from (the old) d' to q are non-convex, the new points b' and a' are the same as before, and the new d' is the old c'. Thus, we can again switch $b'd'$, and so on, until Q is type 1 or $c' = q$ at which point the next switch decreases the number of edges intersecting pq. This finally contradicts property (1). $\square$

5.3 Preliminaries Results on rng-Polygons

This section introduces some notations, and presents a few elementary lemmas for our subsequent discussion to speed up the cubic time algorithm (Section 5.1) to quadratic time by a specialized polygon triangulation algorithm. Recall that the first two steps of the algorithm decompose the convex hull of S into simply connected regions by drawing all edges of $ch(S)$ and $rng(S)$; these steps remain unaltered. Each region is represented by a polygon, termed an *rng-polygon*, with directed edges that trace the boundary of the region in a counterclockwise order around the region. Because $rng(S)$ is a connected graph that spans S, any rng-polygon has at most one edge not in $rng(S)$; this edge is in $ch(S) - rng(S)$. We call an rng-polygon a *complete rng-polygon* if all its edges belong to $rng(S)$, and an *incomplete rng-polygon*, otherwise.

Obviously, rng-polygons are not as general as arbitrary polygons because for each edge ab, except possibly for one, the lune of ab, $\lambda_{ab} = \{x \in \mathbb{R}^2 : \max\{|ax|, |bx|\} < |ab|\}$, is free of points of S. We call pq a *diagonal* of an rng-polygon if it lies entirely in the interior of the rng-polygon. For each diagonal pq of an rng-polygon it must be that λ_{pq} contains at least one point of S. We further distinguish between the cases where λ_{pq} contains points of S on both sides of the line through pq and where it does not.

For a directed edge $\vec{pq}$, let $h_{\vec{pq}}$ be the set of points to the left of or on the directed line that passes through p and q in this order. Define the *half-lune* of $\vec{pq}$ as

$$\eta_{\vec{pq}} = \lambda_{pq} \cap h_{\vec{pq}}.$$

By definition, $\lambda_{pq} = \eta_{\vec{pq}} \cup \eta_{\vec{qp}}$, and we have $pq \in rng(S)$ iff $\eta_{\vec{pq}} \cap S = \eta_{\vec{qp}} \cap S = \emptyset$. We call pq an *rng-edge* if $pq \in rng(S)$, call it a *1-edge* if only one half-lune contains points of S, and call it a *2-edge* if both half-lunes contain points of S. For a 1-edge pq, we say the side where the half-lune contains points of S is *beyond* pq, and the other side is *beneath* pq. Note for example that if pq is a 1-edge bounding an incomplete rng-polygon R, then $pq \in ch(S)$ and therefore R is beyond pq. We will see later that 1-edges are useful in triangulating rng-polygons.

The first lemma of this section shows that when we triangulate an rng-polygon R, whether complete or incomplete, we can ignore all points outside R. More specifically, it shows that the type of any diagonal or edge of R remains unchanged when we remove all points of S that are not vertices of R.

Reduction Lemma. Let pq be a diagonal or edge of an rng-polygon R. If $\eta_{\vec{pq}}$ contains points of S, then it also contains vertices of R.

Proof. Suppose $\eta_{\vec{pq}}$ contains points of S but no vertices of R. Then, it must intersect edges of R without containing their endpoints. Let yy' be the edge closest to p and q, and let x be a point in $\eta_{\vec{pq}} \cap S$. Since x is not a vertex of R it must lie on the other side of yy', as seen from p and q. So $yy' \in rng(S) - ch(S)$, and therefore $\max\{|xy|, |xy'|\} \geq |yy'|$. Assume without loss of generality that $|xy| \geq |yy'|$. If y' lies outside or on the circle $(p, |pq|)$, we consider the convex quadrilateral $pyxy'$. Otherwise, y' lies outside or on $(q, |pq|)$, in which case we consider the convex quadrilateral $qyxy'$. But now we have $|xy| \geq |yy'|$ and either $|py'| > |px|$ or $|qy'| > |qx|$, a contradiction to the $\square$-Lemma in both cases. $\square$

Using the Reduction Lemma, we now address vertices visible from both endpoints of an edge. We need some terminology. Two points x, y inside or on the boundary of an rng-polygon are *visible* from each other if xy is contained in the interior of the rng-polygon. The *distance* of a point x to an edge pq is defined as the infimum, over all points $z \in pq$, of $|xz|$. If $|pq| > \max\{|px|, |qx|\}$, then this distance is indeed the height of triangle pqx.

Visibility Lemma. Let pq be a diagonal or edge of an rng-polygon R, and let x be a vertex of R that lies in $\eta_{\vec{pq}}$ and minimizes the distance from pq. Then x is visible from p and also from q.

Proof. Consider the triangle pqx, let $x' \in pq$ be the point with minimum distance from x, and assume without loss of generality that x is not visible from q. Let yy' be an edge of R that intersects qx. The proof of the Reduction Lemma implies that at least one endpoint of yy' lies in $\eta_{\vec{pq}}$, say $y \in \eta_{\vec{pq}}$. In addition, y and y' lie outside pqx because x is closest to pq (see Figure 5.8). Hence, yy' intersects xp, xq and all edges xz with $z \in pq$. Thus, $xyx'y'$ is a convex quadrilateral, and because of $|yx'| \geq |xx'|$ by the choice of x, we have $|yy'| > |xy'|$ from

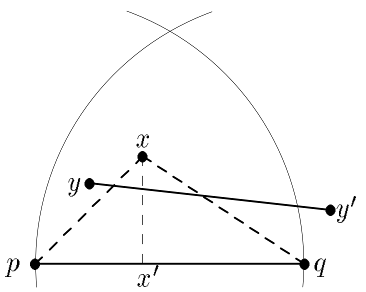

Figure 5.8: The quadrilateral $xyx'y'$ is convex because $x' \in pq$ and $y, y' \notin pqx$.

the $\square$-Lemma. By symmetry, if y' lies in $\eta_{\vec{pq}}$ we have $|yy'| > |xy|$, which implies $yy' \notin rng(S)$. This is a contradiction because $yy' \notin ch(S)$. Thus, y' must lie outside $\eta_{\vec{pq}}$. If y' lies outside or on the circle $(p, |pq|)$, then $|py'| > |px|$, and therefore $|xy| < |yy'|$ by the $\square$-Lemma for $py'xy$. Symmetrically, we get $|xy| < |yy'|$ from the $\square$-Lemma for $qy'xy$ if y' lies outside or on the circle $(q, |pq|)$. Together with $|xy'| < |yy'|$, this contradicts $yy' \in rng(S)$. $\square$

We need one more elementary lemma.

Containment Lemma. If $x \in \eta_{\vec{pq}}$, then $\eta_{\vec{xp}} \subseteq \lambda_{pq}$.

Proof. Take a point $z \in \eta_{\vec{xp}}$ and consider the four points p, q, x, z. If $z \in pq$, there is nothing to prove. Otherwise, $pzqx$ or $pqzx$ is a convex quadrilateral (possibly with three of the four vertices collinear) or $z \in pqx$. In each case $|qz| < |pq|$ can be shown using the $\square$- or the Δ-Lemma. This implies $z \in \lambda_{pq}$. $\square$

The following lemma is of fundamental importance to the quadratic time triangulation algorithm.

1-Edge Lemma. Let pq be a 1-edge of an rng-polygon R, and let x be a vertex of R that lies in $\eta_{\vec{pq}}$ and minimizes the distance from pq. Then px is either an edge of R or a 1-edge with pqx beneath px, and the same is true for qx.

Proof. We have $\eta_{\vec{xp}} \subseteq \lambda_{pq}$ by the Containment Lemma. The part of $\eta_{\vec{xp}}$ in $\eta_{\vec{qp}}$ contains no point of S because $\eta_{\vec{qp}} \cap S = \emptyset$ by assumption. Also, the part of $\eta_{\vec{xp}}$ in $\eta_{\vec{pq}}$ also contains no point of S because a point $y \in \eta_{\vec{xp}} \cap \eta_{\vec{pq}}$ would be closer to pq than x is, as can be shown using the $\square$-Lemma for $px'yx$ (see Figure 5.9). So px is an edge of R if $\eta_{\vec{px}}$ contains no point of S either, and it is a 1-edge with triangle pqx on its beneath side, otherwise. The argument for qx is symmetric. $\square$

5.4 Triangulating Incomplete rng-Polygons

The above lemmas are sufficient for efficiently triangulating an incomplete rng-polygon. As defined earlier, all edges of an incomplete rng-polygon R are rng-edges, except for one 1-edge,

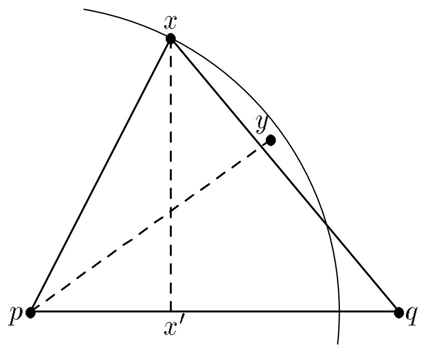

Figure 5.9: Vertex x is visible from p and from q, so pqx is empty. It follows that if $y \in \eta_{\vec{xp}} \cap \eta_{\vec{pq}}$, then $pqyx$ is a convex quadrilateral.

$pq \in ch(S) - rng(S)$, which has R on its beyond side. The algorithm below can triangulate more general incomplete rng-polygons, that is, it is not necessary that $pq \in ch(S)$, but it must be that pq is a 1-edge and R lies beyond pq.

Input. An incomplete rng-polygon R that lies beyond its 1-edge pq.

Output. A min-max length triangulation of R.

Algorithm.
1. Find a vertex x in λ_{pq} that minimizes the distance from pq.
2. Draw edges px and qx. This decomposes R into the triangle pqx, and two possibly empty incomplete rng-polygons R_1 and R_2.
3. Recursively triangulate R_1 and R_2.

The correctness of this algorithm follows from the 1-Edge Lemma. Indeed, it implies that if R_1 is non-empty, then it lies beyond px, which is the only 1-edge of R_1. Similarly, R_2 lies beyond its 1-edge qx, provided R_2 is non-empty. Thus, the input invariant is maintained all the way through the recursion. This implies that the algorithm successfully triangulates. By the choice of point x, the edges px and qx are both shorter than pq. It follows that the diagonals are monotonely decreasing in length, down a single branch of the recursion, and therefore all diagonals constructed by the algorithm are shorter than pq. A straightforward implementation of the algorithm takes time quadratic in the number of vertices of R.

Remark. Instead of choosing a vertex x that minimizes the distance to pq, step 1 of the algorithm could also choose other vertices as long as they are visible from p and q and lie in their lune. An interesting choice among these vertices is the vertex y that minimizes $\max\{|yp|, |yq|\}$. As long as y is unique, which is the non-degenerate case, this choice leads to a triangulation of the polygon R that lexicographically minimizes the non-increasing vector of edge lengths. Another possible choice is the vertex z that minimizes $|zp| + |zq|$. This vertex is automatically visible from p and from q and might be useful in actual implementations because it is often considerably less expensive to compute the distance between two points than between a point and a line segment.

5.5 A Lemma on Polygon Retriangulation

This section presents a technical lemma on retriangulating a polygon. It will find application in the next section, and is also of independent interest. In order to conveniently distinguish between boundary and non-boundary edges of a triangulation, we call a non-boundary edge a *diagonal*. Let X be a polygon, $\mathcal{X}$ a triangulation of X, and xx' a diagonal of X that is not in $\mathcal{X}$. We say that xx' *generates* $\mathcal{X}$ if it intersects every diagonal of $\mathcal{X}$. We give an algorithmic description of a particular triangulation of X, called the *fan-out triangulation* $\mathcal{F}_x(X)$ with *(fan-out) center* x. The triangulation is illustrated in Figure 5.10.

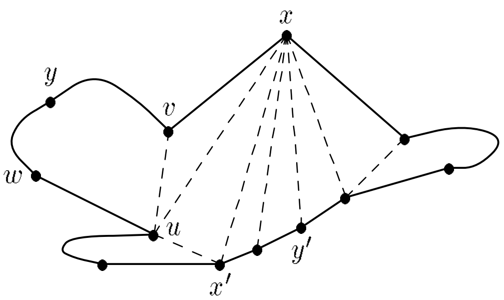

Figure 5.10: The polygon X is triangulated by fanning out from x, connecting adjacent neighbors of x, and recursing in the thus created pockets. The illustration of this process is schematic and ignores some of the inherent shape constraints for X.

1. Connect x to all vertices of X that are visible from x. Call these vertices and also the two vertices connected to x by edges of X *neighbors* of x.

2. Two neighbors of x are said to be *adjacent* if they are consecutive in the angular order around x. Connect any two adjacent neighbors u, v of x, unless uv is an edge of X.

3. Every edge uv created in step 2 decomposes X into two parts, and the part that does not contain x is called the *pocket* X_{uv} of uv. Assume that u is the endpoint of uv so that the

other incident edge of the pocket, uw, is partially visible from x. Recursively construct the fan-out triangulation of X_{uv} with center v.

We need a few more terminology. Among the diagonals of $\mathcal{F}_x(X)$, we distinguish between *fan-out* edges constructed in step 1 and *cut-off* edges constructed in step 2 of the above algorithm. Each call of the algorithm triangulates part of a pocket and recurses in each component (pocket) of the remainder. We call a pocket V a *child* of another pocket Z if $V \subset Z$ and V is maximal. The original polygon, X, is also called a pocket and forms the root of the tree defined by the child relation. This tree is exactly the recursion tree of the algorithm. Each pocket Z is associated with a fan-out center z. The maximum distance between z and any other vertex of Z is called the *width* of Z.

The lengths of the diagonals of $\mathcal{F}_x(X)$ are constrained by the length of the longest edge of X, the length of the longest diagonal of $\mathcal{X}$, and the width of X. Specifically, we prove the following result.

Fan-Out Lemma. Let X be a polygon, with δ_1 the length of its longest edge, let $\mathcal{X}$ be a triangulation of X, with δ_2 the length of its longest diagonal, let xx' be a generator of $\mathcal{X}$, and let δ_3 exceed the maximum distance of x from any vertex of X. Then $|ab| < \max\{\delta_1, \delta_2, \delta_3\}$ for every diagonal ab of $\mathcal{F}_x(X)$.

Proof. Note that the assertion follows if we prove that $\max\{\delta_1, \delta_2, \delta_3\}$ exceeds the width of every pocket Z created during the algorithm. To see this, notice that the width of Z is an upper bound on the length of any fan-out edge emanating from the center of Z. Each cut-off edge uv that creates a child pocket V of Z is incident to the fan-out center of V, which implies that the width of V is an upper bound on its length.

The proof of the upper bound on the widths of all pockets proceeds inductively, from the top to the bottom of the tree. The width of X is less than δ_3, by assumption, and therefore also less than $\max\{\delta_1, \delta_2, \delta_3\}$. For the inductive step, consider a pocket Z and a child V of Z. We show that the bound on the width of Z is inherited by V, with some environmental influence from X and $\mathcal{X}$. Let z be the fan-out center of Z, δ the width of Z, v the fan-out center of V, uv the cut-off edge that creates V, and w the other vertex of V adjacent to u.

First, we prove $|uv| < \max\{\delta_1, \delta\}$. By definition of fan-out center, v lies in the closure of uwz. The Δ-Lemma thus implies $|uv| < \max\{|uw|, |uz|\}$, and we get the claimed inequality because $|uw| \le \delta_1$ and $|uz| \le \delta$. Second, we show that $\max\{\delta_2, \delta\}$ exceeds the maximum distance between v and any vertex of V other than u. Let $y \ne v, u$ be such a vertex and let yy' be a diagonal of $\mathcal{X}$ that intersects xx'. Such a diagonal exists because xx' generates $\mathcal{X}$. It follows that yy' intersects uv and that therefore v lies in the closure of $yy'z$. Using the Δ-Lemma, we get $|yv| < \max\{|yy'|, |yz|\} \le \max\{\delta_2, \delta\}$ because $|yy'| \le \delta_2$ and $|yz| \le \delta$. The two bounds together imply that the width of V is less than $\max\{\delta_1, \delta_2, \delta\}$, and induction shows that it is less than $\max\{\delta_1, \delta_2, \delta_3\}$. □

In Section 5.6.3, we will need a result as given in the Fan-Out Lemma, but restricted to the fan-out triangulation on one side of the generator. Specifically, we need the following corollary whose proof is almost the same as the one of the Fan-Out Lemma.

Fan-Out Corollary. Suppose W is a polygon, $\mathcal{W}$ a triangulation of W, xx' a generator of $\mathcal{W}$, and X the part of W on one side of xx'. Let δ_1 be the length of the longest edge of X, δ_2 the length of the longest diagonal of $\mathcal{W}$, and let δ_3 exceed the maximum distance of x from any vertex of X. Then $|ab| < \max\{\delta_1, \delta_2, \delta_3\}$ for every diagonal ab of $\mathcal{F}_x(X)$.

Remark. The Fan-Out Lemma can also be formulated without the assumption of an initial triangulation. The condition on the diagonal xx' is now that each vertex of X must be visible from some point of xx'. The parameter δ_2 needs to be redefined as the maximum, over all vertices y of X, of the infimum, over all points a of xx' visible from y, of the distance between y and a.

5.6 Triangulating Complete rng-Polygons

This section shows how to triangulate a complete rng-polygon R in quadratic time. The algorithm is given in Section 5.6.1. The basic idea is to divide R into two polygons, each can be triangulated by the algorithm for incomplete rng-polygons (Section 5.4). This idea is supported by the 2-edge Lemma in Section 5.6.2. That is, when vertices are in general position, there exists a min-max length triangulation of R that contains *exactly one* 2-edge, say pq. Moreover,

pq indeed divides R into two polygons that behavior as required, and the resulting triangulation thus has pq as its longest edge. Such a 2-edge can, fortunately, be computed in quadratic time as shown in Section 5.6.3. The algorithm thus has the claimed time-complexity.

5.6.1 The Algorithm

We assume that no two diagonals and edges of the *rng*-polygon R are equally long. If this non-degeneracy assumption is not satisfied, it is necessary to run the algorithm with a simulation of non-degeneracy. The side-effects of this simulation and how they can be undone will be discussed in Section 5.7.

Let us call a 2-edge pq *expandable* if there are vertices r and s in λ_{pq}, on different sides of the line through pq and both visible from p and q, so that $E = \{pr, qr, ps, qs\}$ is a set of *rng*- and 1-edges and the quadrilateral $prqs$ lies beneath the 1-edges in E. The algorithm as shown computes the shortest expandable 2-edge pq (steps 1 to 3), and then completes the triangulation using the algorithm for incomplete *rng*-polygons (step 4). The resulting triangulation uses no 2-edge other than pq, which is thus the longest edge of the triangulation.

Input. A complete *rng*-polygon R.

Output. A min-max length triangulation of R.

Algorithm.
1. Determine the type of each diagonal pq of R.
2. For each 2-edge pq, find vertices p', p'', q', q'' that minimize the counterclockwise angles $\angle p'pq, \angle qpp'', \angle q'qp, \angle pqq''$, contingent upon pp', pp'', qq', qq'' being *rng*-edges or 1-edges with pq on their beneath sides (see Figure 5.11).
3. Find the shortest 2-edge pq for which pp', qq', pp'', qq'' are such that $p' = q''$ or $pp' \cap qq'' \neq \emptyset$, and $p'' = q'$ or $pp'' \cap qq' \neq \emptyset$.
4. As pq is expandable (Section 5.6.3), find $E = \{pr, qr, ps, qs\}$ and then triangulate the (possibly empty) incomplete *rng*-polygons defined by pr, qr, ps, qs.

Below we give the algorithmic details of the above steps.

<u>**Step 1**</u>: *classifying diagonals.* For each vertex p of R, we compute all incident diagonals pq and their angular order around p. Furthermore, we determine whether or not the half-lune $\eta_{\vec{pq}}$

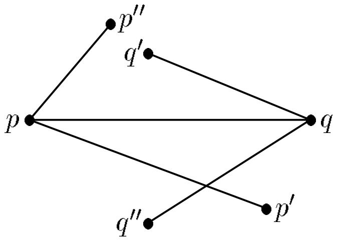

Figure 5.11: By the choice of p' the counterclockwise angle $\angle p'pq$ contains no 1-edge with pq on its beneath side. Symmetric statements hold for p'', q', and q''.

contains any vertex of R. Recall that, by the Visibility Lemma, $\eta_{\vec{pq}}$ contains a vertex visible from p if it contains a vertex of R at all. We can thus base the decision whether or not $\eta_{\vec{pq}}$ is empty of vertices solely on the vertices visible from p. As defined earlier, pq is a 2-edge if both half-lunes of pq contain vertices of R. Otherwise, pq is a 1-edge and its beyond side is where the half-lune contains vertices of R. We now show that the computation for p can be done in time linear in the number of vertices. It follows that quadratic time suffices for step 1.

Computing the sorted sequence of diagonals $pp_1, pp_2, \ldots, pp_m$ incident to p is a standard operation for simple polygons and can be done in linear time; see, for example, [ElAv81, JoSi87, Lee83]. Let pp_0 and pp_{m+1} be the two edges of R incident to p and assume that $p_0, p_1, p_2, \ldots, p_{m+1}$ is in a counterclockwise order around p.

To determine whether there is a vertex of R in the half-lune $\eta_{\vec{pp_i}}$ for $1 \leq i \leq m$, we scan the list $p_0, p_1, \ldots, p_{m+1}$ once, from smallest index to largest. During the scan, we maintain a stack of diagonals pp_l whose half-lunes $\eta_{\vec{pp_l}}$ are not yet found to contain any vertex of R. Before pushing pp_i onto the stack, we remove all diagonals pp_l whose half-lunes contain p_i. Using a straightforward extension of the Containment Lemma, we can show that the order of processing implies that the edges whose half-lunes contain p_i lie on top of the ones whose half-lunes do not contain p_i. Thus, the former can be removed simply by repeatedly popping the topmost diagonal. When the scan is complete, the stack contains exactly all diagonals pp_l whose half-lunes contain no vertex of R. Since a diagonal can be pushed and popped only once each, the entire process takes constant time per diagonal.

<u>STEP 2</u>: *finding rng- and 1-edges.* For each vertex p, we scan $pp_1, pp_2, \ldots, pp_m$ in this order. In the process, we keep track of the most recent rng-edge or 1-edge $p\bar{p}$ whose beneath side is in the direction of the scan. Initially, $p\bar{p} = pp_0$. When a 2-edge pq is encountered, then $p\bar{p}$ is the

edge pp' that belongs to pq. A symmetric scan is carried out to find the edge pp'' that belongs to pq. The total time, for all vertices p of R, is clearly quadratic.

STEP 3: *locating shortest expandable 2-edge.* Step 3 is computationally trivial. It takes quadratic time since constant time suffices to test whether or not pp', pp'', qq', qq'' satisfy the conditions of step 3. However, it is not trivial to see that the edge pq obtained by step 3 is also the shortest expandable 2-edge. We defer the proof till Section 5.6.3.

STEP 4: *triangulating incomplete rng-polygons.* It should be clear that we can simply scan through the vertices of R once to obtain E, then apply the algorithm for incomplete *rng*-polygons (Section 5.4). This step takes time only quadratic in the number of vertices of R. Incidentally, pq is in fact a 1-edge when restricted to only one of the two polygons. So, the step can actually be simplified to only two applications of the algorithm for incomplete *rng*-polygons.

The next two subsections show the correctness of the above procedure.

5.6.2 Triangulating by One 2-Edge

Under the non-degeneracy assumption, we note that every triangulation of R, and therefore also every min-max length triangulation, contains a 2-edge. To see this, consider the longest edge pq of a triangulation. It cannot be an edge of R because the third vertex of the incident triangle lies in its lune λ_{pq}. It is therefore a diagonal with incident triangles pqr and pqs, and we have $r, s \in \lambda_{pq}$ by maximality of pq. Since r and s lie on different sides of the line through pq, it follows that pq is indeed a 2-edge. In fact, we have the following stronger result utilized by the above algorithm.

2-Edge Lemma. Let R be a complete *rng*-polygon with no two diagonals or edges of the same length. Then there exists a min-max length triangulation of R that contains an expandable 2-edge.

Proof. We assume there is no min-max length triangulation of R that contains an expandable 2-edge. A contradiction to this assumption will be derived using an min-max length triangulation of R, $\mathcal{R}$ defined as follows. Let pq be the longest edge of $\mathcal{R}$ and let pqr and pqs be the incident triangles. By the non-degeneracy assumption, pq is the longest edge of *every* min-max length triangulation of R. Choose $\mathcal{R}$ so that the sum of heights of pqr and pqs (that is, the distance of r from pq plus the distance of s from pq) is a minimum. We prove below that pq is expandable and that r and s are witnesses thereof, that is, the quadrilateral $prqs$ lies beneath every 1-edge in $E = \{pr, qr, ps, qs\}$.

Case 1. Assume that $prqs$ lies beyond at least one 1-edge in E, say beyond pr. Then we can retriangulate R on this side of pr using the algorithm for incomplete rng-polygons. Among others, this algorithm removes edge pq, and all new edges are shorter than pr, which itself is shorter than pq. This contradicts the assumption that $\mathcal{R}$ is a min-max length triangulation.

Case 2. Assume that one of the edges of E, say pr, is a 2-edge, and assume without loss of generality that $r \in \eta_{\vec{pq}}$. Thus, there is a non-empty set of vertices z of R contained in the half-lune $\eta_{\vec{rp}}$. By the Containment Lemma, these vertices z lie in λ_{pq}, and by the Visibility Lemma, a non-empty subset S' of the z are visible from both p and r.

If a vertex z is in S', then either $pz \cap rq \neq \emptyset$ or $rz \cap pq \neq \emptyset$; see Figure 5.12. Let S'_p be

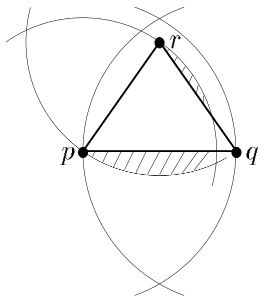

Figure 5.12: The points z lie in the interior of $\eta_{\vec{rp}} - pqr$, which consists of one or two connected components depending on whether or not the angle at r in pqr is non-acute.

the subset of vertices z of the first kind, and let S'_r be the subset of vertices of the second kind. If $S'_p \neq \emptyset$, choose $x \in S'_p$ so that the number of edges of $\mathcal{R}$ that intersect px is a minimum. Next, remove all edges from $\mathcal{R}$ that intersect px and denote by X the polygon thus generated. If, on the other hand, $S'_p = \emptyset$, then choose $x \in S'_r \neq \emptyset$ so that the number of edges in $\mathcal{R}$ that intersect rx is a minimum; again remove all edges from $\mathcal{R}$ that intersect rx, and denote the resulting polygon by X. For convenient reference, we set $x' = p$ in the first case and $x' = r$ in the second. In either case, we construct a retriangulation $\mathcal{F}_x(X)$ of X by fanning out from x, as described in Section 5.5.

We show below that the new triangulation of R has properties that contradict the assumptions of case 2. Most importantly, the Fan-Out Lemma of Section 5.5, together with a few claims which we are about to prove, imply that the edges of $\mathcal{F}_x(X)$ do not exceed pq in length.

Claim 1. Except for x, all vertices of X lie outside the half-lune $\eta_{\vec{rp}}$.

Proof (of Claim 1). Let $y_1y_2, y_3y_4, \ldots, y_{m-1}y_m$ be the edges, sorted from x' to x, that are removed from $\mathcal{R}$ when X is constructed. (y_1y_2 is either pq or qr, and $y_3 \in \{p, q, r\}$.) Suppose the claim is not true. Then there is a smallest index j, $3 \leq j \leq m-1$, so that one endpoint of y_jy_{j+1}, say y_{j+1}, is in $\eta_{\vec{rp}}$. Consider the polygon X_j of $\mathcal{R}$ that is created by removing the edges $y_1y_2, y_3y_4, \ldots, y_{j-2}y_{j-1}$ from $\mathcal{R}$. Since y_{j+1} is the only vertex of X_j that lies in $\eta_{\vec{rp}}$, it is visible from p *and* from r inside X_j. But this means that $y_{j+1}x'$ intersects fewer edges of $\mathcal{R}$ than xx'. This contradicts the choice of x and completes the proof of Claim 1.

Claim 2. For each vertex y of X, we have $|xy| < |pq|$.

Proof (of Claim 2). Clearly, both px and rx are shorter than pq. So let y be any vertex different from p, r, x, and let yy' be an edge of $\mathcal{R}$ that intersects $x'x$. Because of Claim 1, x is visible within X from p and also from r, so $pyxy'$ and $ryxy'$ are convex quadrilaterals. Since y' lies outside $\eta_{\vec{rp}}$, it cannot lie inside both of the circles $(p, |pr|)$ and $(r, |pr|)$. If y' lies inside $(r, |pr|)$, then $|py'| > |px|$ which implies $|yy'| > |xy|$ by the $\square$-Lemma for $pyxy'$. Otherwise, we have $|ry'| > |rx|$ which implies $|yy'| > |xy|$ by the $\square$-Lemma for $ryxy'$. This concludes the proof of Claim 2 because yy' is an edge of $\mathcal{R}$ and is therefore no longer than pq.

Claim 2 and the Fan-Out Lemma imply that all diagonals of $\mathcal{F}_x(X)$ are shorter than pq. In the case where $pq \cap rx \neq \emptyset$, we now have a contradiction, because the retriangulating process of X eliminates pq and all edges of the resulting new triangulation of R are shorter than pq. In the case where $rq \cap px \neq \emptyset$, the new triangulation still includes pq. We show below that the height of the new triangle incident to pq is smaller than the height of pqr, and thus arrive at a contradiction.

So assume $rq \cap px \neq \emptyset$; in this case pq is an edge of the boundary of X and p is visible from x. If q is also visible from x, then the new triangle incident to pq is pqx with height $|xx'|$, where $x' \in pq$ minimizes the distance to x. Analogously, define $r' \in pq$ that minimizes the distance to r. Since $|pr| > |px|$, we have $|rr'| > |xr'|$ by the $\square$-Lemma for $prxr'$. Together with $|xr'| \geq |xx'|$, this implies $|rr'| > |xx'|$. If q is not visible from x, then pq belongs to the pocket X_{uv} defined by a cut-off edge uv. We have $u = p$, $w = q$, and the center v of X_{uv} lies in the closure of pqx. So again, either pqv is a triangle, and its height is less than that of pqx and therefore that of pqr, or q is not visible from v, in which case the argument can be repeated. Eventually, we arrive at a triangle incident to pq whose height is less than that of pqr. $\square$

Two remarks are in order. First, the expandable 2-edge mentioned in the lemma is clearly the shortest among all expandable 2-edges. Second, the assertion of the Lemma is false without the condition that no two diagonals or edges of the complete rng-polygon R are equally long. To see this, take, for example, two equilateral triangles abc and abd and move d slightly towards the common edge ab. For $S = \{a, b, c, d\}$, we have $rng(S) = \{ac, cb, bd, da\}$, ab is a 1-edge, and cd is a 2-edge. So $acbd$ is a complete rng-polygon. There is only one min-max length triangulation of $acbd$, namely, the one with diagonal ab. But ab is not a 2-edge.

5.6.3 Searching for the Right 2-Edge

Two results are needed to show that the 2-edge pq computed by step 3 in Section 5.6.1 is indeed the shortest expandable 2-edge. First note that there are no expandable 2-edges shorter than pq. This is because all 2-edges shorter than pq fail the test of step 3, which are thus not expandable as implied by the following topological lemma.

Crossing Lemma. Let $v_1, v_2, \ldots, v_n$ be the sequence of vertices of a simple polygon, and let v_1v_i and v_jv_n be two diagonals. Then $v_1v_i \cap v_jv_n \neq \emptyset$ iff $j < i$.

Proof. The edge v_jv_n decomposes the polygon into two polygons with vertex sequences $v_1, v_2, \ldots, v_j, v_n$ and $v_j, v_{j+1}, \ldots, v_n$. If $j < i$, then neither of the two polygons has v_1 *and* v_i on its boundary. It follows that v_1v_i crosses from one polygon into the other, and because v_1v_i is a diagonal, this is only possible by crossing v_jv_n. To prove the other direction, we assume $v_1v_i \cap v_jv_n \neq \emptyset$ and observe that v_1 and v_i belong to different polygons because there is no way that v_1v_i can enter the second polygon and leave it again. Thus, $j < i$. $\square$

We remain to show that the edge pq computed in step 3 is indeed expandable.

Expandability Lemma. The shortest 2-edge pq of R that satisfies the conditions of step 3 is also expandable.

Proof. We show below that R can be triangulated on both sides of pq using only edges shorter than pq. If we now assume that pq is not expandable, we get a contradiction to the 2-Edge Lemma because pq is the longest edge of the triangulation and all expandable 2-edges are longer than pq.

We describe how to triangulate the part of R to the right of $\vec{pq}$; the other part is symmetric.

Case 1: $p' = q''$. Assume $|qq''| > |pp'|$. Then $|qq''| < |pq|$ for otherwise $p \in \eta_{\overrightarrow{q''q}}$ and qq'' would neither be an rng-edge nor a 1-edge with pq on its beneath side. If we apply the triangulation algorithm for incomplete rng-polygons (Section 5.4), once for pp' and once for qq'', we get a triangulation with the desired properties.

Case 2: $pp' \cap qq'' \neq \emptyset$. In this case pp' and qq'' are 1-edges. Because pp' and qq'' intersect, we have either $|p'q| < |p'p|$ or $|q''p| < |q''q|$ from the $\Box$-Lemma for $pqp'q''$. Assume without loss of generality that $|q''p| < |q''q|$. As in case 1 we also have $|q''q| < |pq|$, but note that we do not necessarily have $|p'p| < |pq|$.

We now describe the triangulation process. It takes three steps as illustrated in Figures 5.13 and 5.14.

1. Construct the triangulation $\mathcal{T}_{qq''}$ of R beyond qq'', using the algorithm for incomplete rng-polygons (see Figure 5.13).

2. Find the subset V of vertices of R, excluding q, that lie in the closure of pqq'', and compute $ch(V)$. Add the edges of $ch(V)$ that are diagonals of R to the triangulation, and connect q to all vertices of $ch(V)$ (see Figure 5.13).

3. Step 2 created untriangulated pockets Y_{uv}, one for each edge uv of $ch(V)$ that is a diagonal of R. Assume that u precedes v on the clockwise path from p to q'' on the boundary of $ch(V)$. The pocket Y_{uv} is triangulated as follows.

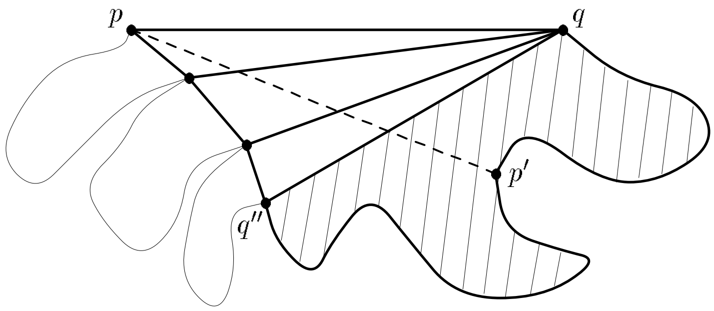

Figure 5.13: The shaded portion represents the triangulation beyond qq''; it forms part of the final triangulation.

3.1 Set $u_L := v$ if uv is a 1-edge and pq lies on the beneath side of uv. Otherwise, find a vertex u_L so that $|uu_L| < |pq|$, uu_L is a 1-edge, pq lies beneath uu_L, and uu_L does not intersect $ch(V)$. (The existence of such a vertex u_L will be established shortly.)

3.2 Construct the triangulation $\mathcal{T}_{uu_L}$ of R beyond uu_L, again using the algorithm for incomplete *rng*-polygons, but retain only those triangles that lie completely inside the pocket Y_{uv}. Let X_{uv} denote the untriangulated part of Y_{uv}.

3.3 Construct the fan-out triangulation $\mathcal{F}_v(X_{uv})$.

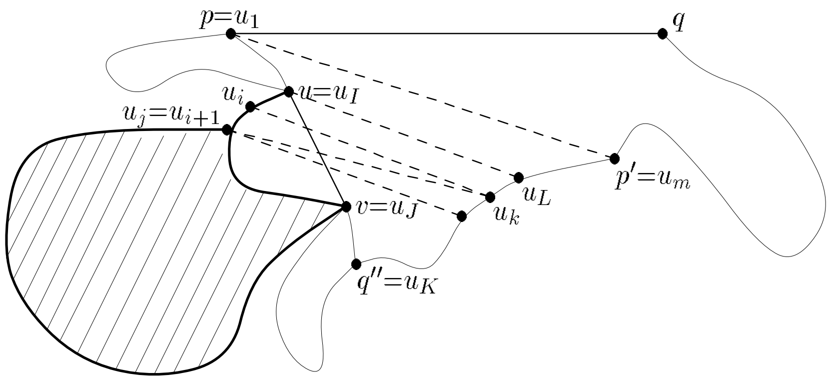

Figure 5.14: The shaded portion of the pocket Y_{uv} represents the part of the triangulation $\mathcal{T}_{uu_L}$ beyond uu_L that is retained for the final triangulation. The remaining portion is triangulated by fanning out from v.

The remainder of the proof establishes that all diagonals of the thus constructed triangulation are shorter than pq. This is indeed obvious for $\mathcal{T}_{qq''}$, as constructed in step 1. We now prove an easy extension of the Δ-Lemma, which implies that all edges created in step 2 are shorter than pq.

Claim 1. Let d, e be two points in the closure of triangle abc. Then $|de| < \max\{|ab|, |ac|, |bc|\}$.

Proof (of Claim 1). Assume without loss of generality that e lies in the closure of abd. The Δ-Lemma for abd implies $|de| < \max\{|ad|, |bd|\}$, and the same lemma for abc implies $\max\{|ad|, |bd|\} < \max\{|ab|, |ac|, |bc|\}$. This completes the proof of Claim 1.

If $uu_L = uv$, then $|uu_L| < |pq|$, which implies that all edges of $\mathcal{T}_{uu_L}$ constructed in step 3.2 are shorter than pq. In this case the proof is complete as $X_{uv} = \emptyset$ and no edges are added to Y_{uv} in step 3.3. For the remainder of the proof, we thus assume that $u_L \neq v$ which is the case only if $\eta_{\vec{uv}}$ contains at least one vertex of R. We show that a vertex u_L satisfying the conditions of step 3.1 indeed exists, and that all edges of the fan-out triangulation $\mathcal{F}_v(X_{uv})$ are shorter than pq. Assume the sequence of vertices of the part of R beyond pp' is $p = u_1, u_2, \ldots, q'' = u_K, \ldots, u_m = p'$ (see Figure 5.14).

Claim 2. There exists a 1-edge uu_L that satisfies the conditions of step 3.1.

Proof (of Claim 2). Construct a triangulation $\mathcal{T}_{pp'}$ of R beyond pp' using the algorithm for incomplete rng-polygons. This triangulation contains at least one edge uu_l disjoint from $ch(V)$. The main invariant of the algorithm (described in Section 5.4) implies that uu_l is a 1-edge and pq lies on its beneath side. If $|uu_l| < |pq|$, then u_l satisfies the conditions for u_L and we are done.

So assume $|uu_l| > |pq|$. Similar to the Containment Lemma, we can show that the part of $\eta_{\vec{uv}}$ to the left of $\vec{uu_l}$ is contained in $\eta_{\vec{uu_l}}$ and thus contains no vertex of R. It follows that the vertices in $\eta_{\vec{uv}}$ must be among $u_{K+1}, u_{K+2}, \ldots, u_{l-1}$. By the Visibility Lemma, at least one of these vertices is visible from u. Let U be the subset of vertices that are visible from u (including the ones outside $\eta_{\vec{uv}}$), and let $u_L \in U$ minimize the distance to u. We have $|uu_L| < |uv| < |uu_l|$ and, as above, the part of $\eta_{\vec{uu_L}}$ to the left of $\vec{uu_l}$ is contained in $\eta_{\vec{uu_l}}$. Therefore, this part contains no vertex of R. The part of $\eta_{\vec{uu_L}}$ to the right of $\vec{uu_l}$ contains no vertex of R by the choice of u_L. It follows that uu_L is a diagonal that satisfies the conditions of step 3.1, which completes the proof of Claim 2.

We now show two easy facts about $\mathcal{T}_{uu_L}$ before examining the edges constructed by step 3.3.

Claim 3. If $u_iu_ju_k$, with $i < j < k$, is a triangle of $\mathcal{T}_{uu_L}$, then u_iu_k is its longest edge.

Proof (of Claim 3). The first triangle constructed is $u_I u_l u_L$, for some $I < l < L$, and its longest edge is $u_I u_L$ because $u_l \in \lambda_{u_I u_L}$. The general assertion follows by induction, which completes the proof of Claim 3.

Claim 4. The edges of $\mathcal{T}_{uu_L}$ that intersect uv, sorted from u to v, are monotonely decreasing in length.

Proof (of Claim 4). If $u_i u_j u_k$, with $i < j < k$, intersects uv, $u = u_I$ and $v = u_J$, then either $I \leq i < j = i+1 \leq J < k$ or $I < i < J \leq j < k$ (see Figure 5.14). In both cases, $u_i u_k$ intersects uv closer to u than the other intersecting edge, $u_j u_k$ or $u_i u_j$. By Claim 3, $u_i u_k$ is longer than both, which implies the assertion.

Note that if we delete edges from $\mathcal{T}_{uu_L}$ that intersect uv, then we get a polygon, say W_{uv}, of which X_{uv} is the part on one side of uv. We can thus interpret uv as a generator of $\mathcal{T}_{uu_L}$ restricted to W_{uv}. Since the edges of X_{uv} and $\mathcal{T}_{uu_L}$ are shorter than $|pq|$, we just need to show that all vertices of X_{uv} are closer to v than $|pq|$, and the rest follows from the Fan-Out Corollary. Indeed, we prove a stronger bound on the maximum distance from v to a vertex of X_{uv}.

Claim 5. For each vertex x of X_{uv} we have $|vx| \leq |vu|$.

Proof (of Claim 5). Consider the vertices of X_{uv} in turn from $u = u_I$ to $v = u_J$, and assume inductively that $|vu_i| \leq |vu|$, for all $I \leq i < j$. Consider u_j and the triangle $u_{j-1}u_j u_k$ in $\mathcal{T}_{uu_L}$. By Claim 4, we have $|u_{j-1}u_k| > |u_j u_k|$. If $u_{j-1}u_j v u_k$ is a convex quadrilateral, then the $\Box$-Lemma implies $|vu_{j-1}| > |vu_j|$, as desired. Otherwise, u_j is contained in the closure of $vu_k u_{j-1}$ and therefore also in the closure of vuu_{j-1}. The Δ-Lemma implies $|vu_j| < \max\{|vu|, |vu_{j-1}|\}$, which completes the proof of Claim 5.

This also completes the proof of the lemma. $\square$

The following theorem summarizes the algorithmic implication of the above.

Min-Max Length Theorem. A min-max length triangulation of a set of n points in $\mathbb{R}^2$ can be constructed in quadratic time and storage.

5.7 Undoing the Simulated Perturbation

In general, for every point set S in $\mathbb{R}^2$, there is an arbitrarily small perturbation S' so that S' satisfies convenient non-degeneracy assumptions (see [EdMü90]). For a point $p \in S$, we denote its perturbed version by p'. In our algorithm, this means that no two pairs of points in S' define the same distance. Furthermore, the non-degenerate properties of S are maintained, that is, for four not necessarily distinct points $p, q, r, s \in S$ with $|pq| < |rs|$, we have $|p'q'| < |r's'|$.

Let us consider the effects of the perturbation on the computation of a min-max length triangulation. First, if $p'q' \in rng(S')$, then $pq \in rng(S)$, but not vice versa. In other words, we may have fewer edges in the perturbed setting. This, however, does not adversely influence the algorithm since $rng(S')$ is still connected and spans S'. Second, when the edges of $ch(S')$ are added and the polygonal regions defined by $ch(S') \cup rng(S')$ are triangulated, it can happen that triangles $a'b'c'$ are constructed whose unperturbed counterparts abc are flat, that is, a, b, c are collinear. Although this is not a problem for the algorithm, it is somewhat distressing when this triangulation is interpreted as a triangulation of S. The remainder of this section shows how to remedy this deficiency.

Let $\mathcal{T}(S')$ be a min-max length triangulation of S', and consider its unperturbed version $\mathcal{T}(S)$, that is, $pq \in \mathcal{T}(S)$ iff $p'q' \in \mathcal{T}(S')$. A longest edge of $\mathcal{T}(S)$ is no longer than a longest edge of any min-max length triangulation $mlt(S)$ of S, since $mlt(S')$, the perturbed version of $mlt(S)$, is a valid triangulation of S' and would otherwise contradict that $\mathcal{T}(S')$ is a min-max length triangulation of S'. The reverse is also true, namely, a longest edge of $\mathcal{T}(S)$ is no shorter than a longest edge of $mlt(S)$. We show this by converting $\mathcal{T}(S)$ into a min-max length triangulation of S.

Consider the dual graph $\mathcal{T}^*(S')$ of $\mathcal{T}(S')$ and call a node $a'b'c'$ *flat* if a, b, c, are collinear. Determine the connected components of the subgraph of $\mathcal{T}^*(S')$ induced by the set of all flat nodes. Each component corresponds to a collection of collinear points in S, interconnected by flat triangles; see Figure 5.15. Carry out the following steps for one component at a time. Remove all edges of the flat triangles of the component, sort the corresponding points along the supporting line, and add edges connecting points that are adjacent in the sorted order. This produces regions bounded by more than three edges, as shown in Figure 5.15. All vertices

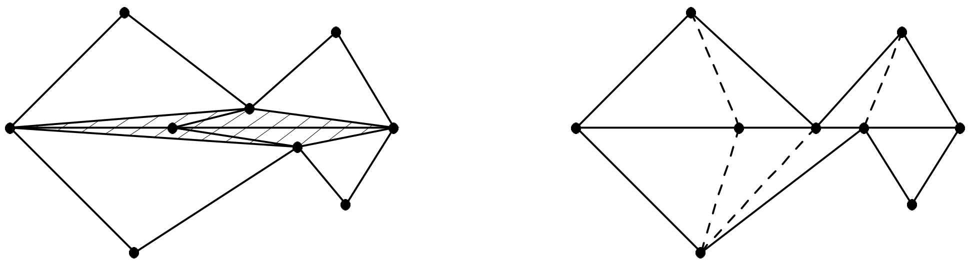

Figure 5.15: The five points in the middle of the left triangulation are the perturbed versions of five collinear points in the right triangulation.

x of such a region are collinear, except for one vertex y, which is connected to the first and last of the vertices x. Triangulate this region by connecting y to all other vertices x. By the Δ-Lemma, the newly introduced edges are no longer than the longer of the two original edges incident to y.

5.8 Extension to Normed Metrics

A convex region $D \subseteq \mathbb{R}^2$ which is symmetric with respect to the origin can be used to impose a *norm* on $\mathbb{R}^2$: for a point $x \in \mathbb{R}^2$, define $\|x\| = \|x\|_D = \alpha$ if x lies on the boundary of $\alpha D = \{\alpha y \in \mathbb{R}^2 : y \in D\}$. The norm can then be used to impose a (*normed*) *metric* on $\mathbb{R}^2$: for two points $x, y \in \mathbb{R}^2$, define $|xy| = |xy|_D = \|y - x\|_D$. D is the *unit-disk* of the metric and the boundary of D is its *unit-circle*. Notice that the three requirements for a metric are indeed satisfied. First, $|ab| = 0$ iff $a = b$ because $\|x\| = 0$ iff x is the origin. Second, $|ab| = |ba|$ because D is centrally symmetric and therefore $\|x\| = \|-x\|$. Third, the triangle inequality, $|ac| \leq |ab| + |bc|$, follows from the convexity of D. Examples of normed metrics are the l_p-metrics, for $p = 1, 2, 3, \ldots$, and the so-called A-metric discussed in [WWW85] for its applications to VLSI.

In this section, we assume that the triangle inequality is strict unless a, b, c lie on a line in this order. This is the case iff the defining convex region D is strictly convex, that is, no line intersects the boundary of D in more than two points. This assumption is convenient and in fact without loss of generality as every convex but not strictly convex region D' can be approximated arbitrarily closely by a strictly convex region D. Computationally, this approximation can be simulated by defining

$$\|x\|_D = \|x\|_{D'} + \varepsilon\|x\|_2,$$

where $\|x\|_2$ is the Euclidean or l_2-norm and ε is an arbitrarily small but positive real number. Clearly, if ε is sufficiently small, then a min-max length triangulation under D is also a min-max length triangulation under D'.

In the remainder of this section, we point out where the developments in Sections 5.1 through 5.6 need to be adjusted when the Euclidean metric is replaced by an arbitrary normed metric. Most importantly, the graphs defined in Section 2.4 can be extended in a natural way. If we now stipulate that "circle" means a homothetic copy of the unit-circle as defined above and "$|ab|$" means the distance under the normed metric defined by D, then the definitions of $mlt(S)$, $\mathcal{D}(S)$, $rng(S)$, and $mst(S)$ can be taken verbatim. The minimum spanning tree, $mst(S)$, is connected and spans S, and the Delaunay triangulation, $\mathcal{D}(S)$, is plane because any two circles intersect in at most two points. Since we still have $mst(S) \subseteq rng(S) \subseteq \mathcal{D}(S)$, we conclude that all three graphs are connected and plane and they span S. We remark that these three graphs are not necessarily plane if D is not strictly convex.

The developments in Sections 5.1 through 5.6 are all based on a small number of basic facts, namely, the distance relations expressed by the $\square$-Lemma and the Δ-Lemma, the convexity of the lune of an edge, and the straightness of the bisector of two points. The $\square$-Lemma and the Δ-Lemma are direct consequences of the triangle inequality and hold in the stated form (with strict inequality) for arbitrary normed metrics as long as D is strictly convex. The lune of two points is clearly convex as it is the intersection of two homothetic copies of D. Unfortunately, the bisector of two points $p \neq q$, $\ell_{pq} = \{x : |xp| = |xq|\}$, is not necessarily straight. Nevertheless, ℓ_{pq} is still a simple curve that divides $\mathbb{R}^2$ into two unbounded regions, called *half-planes*, one containing p and the other q. The two half-planes are star-shaped with respect to p and q, that is, any line through p or q intersects ℓ_{pq} in at most one point. In addition, ℓ_{pq} is symmetric with respect to $\frac{p+q}{2}$ because D is centrally symmetric.

There is only one place where the straightness of the bisector is used in a substantial way, and that is in the proof of Fact 5.3. We restate this fact and show how to prove it without the use of the straightness of the bisector. Recall that bd ($b'd'$) is said to be *switchable* if ac ($a'c'$) is no longer than the longest edge of the triangulation $\mathcal{T}$.

Fact 5.3. It is not possible that both bd and $b'd'$ are non-switchable.

Proof. As established in Fact 5.2, if bd is non-switchable, then a and d are contained in the half-plane defined by ℓ_{pq} that contains q. Symmetrically, if $b'd'$ is not switchable, then a' and d' are contained in the other half-plane. Unlike in the Euclidean case, it is possible that ad and $a'd'$ intersect ℓ_{pq}. It is thus possible that ad precedes $a'd'$ in the order of edges sorted from p to q by their intersections with pq, as in Figure 5.16. Below we argue that if this is the case, then ad (and symmetrically $a'd'$) is switchable. In particular, we show $|ad| > |ap|$ which, together with $|ap| \geq |ac|$ from Fact 5.2, implies that ad is switchable.

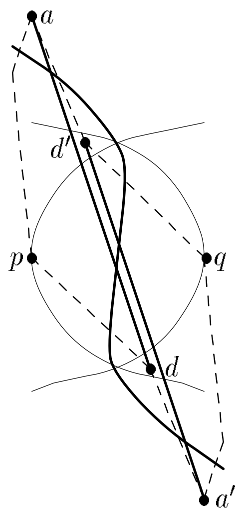

Figure 5.16: Although a and d lie on q's side of the bisector and a' and d' lie on p's side, ad intersects pq closer to p than $a'd'$ does. This is not possible if the bisector is a line as for the Euclidean metric; see Figure 5.6.

One characteristic of the described situation is that ad intersects ℓ_{pq} in at least one point inside the lune of pq. Let x be such an intersection point closest to a. If $pq \cap dx \neq \emptyset$, then $pdqx$ is a convex quadrilateral with $|pd| \geq |pq|$ by construction. The $\square$-Lemma thus implies $|dx| > |qx| = |px|$. It follows that $|ad| = |ax| + |dx| > |ax| + |px| > |ap|$. On the other hand, if $pq \cap dx = \emptyset$, then consider the point $y = ad \cap pq$ and note that $|py| \leq |qy|$. We derive $|dy| > |py|$ from $|py| + |dy| > |pd| \geq |pq| \geq 2|py|$. Therefore, $|ad| = |ay| + |dy| > |ay| + |py| > |ap|$ as desired. $\square$

All other steps of the proof of the Subgraph Theorem go through unchanged for arbitrary normed metrics. We thus get the following generalization.

General Subgraph Theorem. Let S be a point set in $\mathbb{R}^2$ equipped with a normed metric with strictly convex unit-disk. Then S has a min-max length triangulation $mlt(S)$ so that $rng(S) \subseteq mlt(S)$.

So the algorithm for computing a min-max length triangulation is clear—it is the same as for the Euclidean metric, only that the length of edges is now measured in terms of a normed metric. We assume that the length of an edge in this metric can be computed in constant time. A careful reexamination of Sections 5.3 through 5.6 shows that the specialized polygon triangulation algorithm works also in the context of arbitrary normed metrics. We remark, however, that it includes the distance computation between a point and a line segment. Although it is certainly reasonable to assume that this can be done in constant time too, the observation in the remark at the end of Section 5.4 can be used to avoid this computation. We thus have the following algorithmic result which generalizes the Min-Max Length Theorem.

General Min-Max Length Theorem. Let S be a set of n points in $\mathbb{R}^2$ equipped with a normed metric with strictly convex unit-disk. Given the relative neighborhood graph, a min-max length triangulation of S can be constructed in time $O(n^2)$.

As mentioned above, a norm with non-strictly convex unit-disk can be simulated by one with strictly convex unit-disk. It follows that the quadratic time bound also holds for arbitrary normed metrics. The result stated in the General Min-Max Length Theorem raises the question of how fast $rng(S)$ can be constructed. The trivial algorithm tests all $\binom{n}{2}$ edges, each in time $O(n)$, and therefore takes time $O(n^3)$. Faster algorithms are known for the l_p-metrics where $O(n \log n)$ time suffices [JKY92, Kata88, Lee85].

5.9 Discussion

The main contribution of this chapter is the first polynomial time algorithm for computing a min-max length triangulation of a set S of n points in $\mathbb{R}^2$. The components of the algorithm are described in Sections 5.1, 5.4, and 5.6.1. Given the relative neighborhood graph of S, the algorithm takes time $O(n^2)$. The algorithm works for arbitrary normed metrics. The polynomial time bound follows because the relative neighborhood graph of S can be found in polynomial time.

The same problem formulated for general plane geometric graphs can also be solved in polynomial time provided the minimization condition is defined over all edges including the constraining ones. This follows immediately from a constrained version of the Subgraph Theorem. In the Euclidean metric case, the problem can actually be solved in quadratic time because the relative neighborhood graph with constraining edges can be computed in $O(n \log n)$ time [SuCh91]. The question remains whether or not a min-max length triangulation, with or without constraining edges, can be computed in less than quadratic time.

The approach used by the solution is a version of the subgraph approach mentioned in Section 2.5. Both Plaisted and Hong [PlHo87] and Lingas [Ling87] used this approach to compute approximations of the minimum length triangulation. It is interesting to see whether it can also be useful to other criteria. In our case, the technique of *retriangulation* plays an important role in the developments, and this will probably be the case also in other applications of the subgraph approach[1].

The results of this chapter are an out-growth of our general efforts to understand triangulations that optimize length criteria. Currently, min-max length is the only non-trivial one known to be computable in polynomial time. There are many related problems whose complexities remain open; see Section 1.3.2. In particular, the natural extension of the min-max length criterion to its vector form is not solved—it is no longer true that there is always an optimal triangulation that contains the relative neighborhood graph as a subgraph. The smallest example that illustrates this observation consists of four points a, b, c, d so that c and d lie fairly close to b, ab and cd intersect, and c and d both lie outside the circle with radius $|ab|$ and center at a.

[1]Incidentally, retriangulation is also the main idea in proving the correctness of the edge-insertion paradigm.

Acknowledgments

The results in this chapter are jointly developed by Herbert Edelsbrunner and myself [EdTa91]. We thank Jerzy Jaromczyk (University of Kentucky) for pointing out various references on relative neighborhood graph, in particular, the paper by Su and Chang [SuCh91] that implies a quadratic time algorithm for the constrained min-max length triangulation problem in the Euclidean case.

Chapter 6

Conforming Delaunay Triangulations

A conforming Delaunay triangulation for a plane geometric graph $\mathcal{G} = (S, E)$ is introduced in Section 1.2 as an extension to the Delaunay triangulation for a point set. It is a genuine Delaunay triangulation, or more precisely, a completion of a degenerate Delaunay triangulation. Its relation to $\mathcal{G}$ is that each vertex in S is also a vertex of the triangulation, and each edge of E is the union of some edges and vertices of the triangulation [BFL88]. The problem of constructing a conforming Delaunay triangulation for a given plane geometric graph is non-trivial. In particular, it is not clear how many new vertices are really needed to achieve conformity. Previous work [Bois88, NaSr91, Olou91, Saal91, SaPe91] fails to provide polynomial upper bounds on this issue. Such a bound, on the other hand, is given in this chapter.

Section 6.1 formalizes the problem and presents some preliminary results. Section 6.2 gives the first polynomial upper bound on the number of vertices necessary for a conforming Delaunay triangulation. Section 6.3 explicitly formulates the algorithm implicit in the proof of the upper bound. Lastly, Section 6.4 summarizes this chapter and mentions some open problems.

6.1 Preliminaries

Let $\mathcal{G} = (S, E)$ be a plane geometric graph. A completion, $\mathcal{C}$, of a Delaunay triangulation *conforms* to $\mathcal{G}$ if every vertex of $\mathcal{G}$ is a vertex of $\mathcal{C}$ and every edge of $\mathcal{G}$ is the union of edges and vertices of $\mathcal{C}$. The problem is to find a small point set V so that $\mathcal{D}(V)$ has a completion that conforms to $\mathcal{G}$. We call such a completion a *conforming Delaunay triangulation* of $\mathcal{G}$. It is also desirable to have an algorithm that constructs V as well as a completion of $\mathcal{D}(V)$ that conforms to $\mathcal{G}$. The next lemma shows that the latter task can be handled by existing constrained Delaunay triangulation algorithms [Chew89, Seid88], once we have an algorithm that finds V. Thus, we can focus on the process of finding V.

As mentioned before, each edge ab of a completion of $\mathcal{D}(V)$ satisfies the *empty disk property*, that is, there exists a circle through a and b so that no point of V belongs to the disk bounded by the circle. We now argue that this property is also sufficient for the existence of a conforming Delaunay triangulation. Call the portion of an edge of $\mathcal{G}$ between two contiguous points of V (but excluding them) on this edge an *interval*.

Lemma 6.1 $\mathcal{D}(V)$ has a completion that conforms to $\mathcal{G}$ iff every interval defined by $\mathcal{G}$ and V has the empty disk property with respect to V.

Proof. The only if part follows from the definition of a completion of $\mathcal{D}(V)$. For the if part, we assume that every interval ab has the empty disk property. If ab is an edge of $\mathcal{D}(V)$ then nothing has to be proved. Otherwise, no edge of $\mathcal{D}(V)$ can intersect ab because the closure of the disk bounded by any circle passing through the endpoints of such an edge contains a or b or both. Also, there is no other interval that intersects ab because $\mathcal{G}$ is plane. So ab, and all other intervals that are not edges of $\mathcal{D}(V)$, can be added to $\mathcal{D}(V)$ without introducing any intersection. We can add zero or more non-intersecting edges arbitrarily until a completion of $\mathcal{D}(V)$ is obtained. $\square$

<u>A Lower Bound</u>. An edge $ab \in E$ that belongs to the boundary of the convex hull of S automatically satisfies the empty disk property, and no points on ab need to be introduced. For other edges ab, there are points on both sides of the line through ab. It is thus possible that ab

does not satisfy the empty disk property, in which case points must be added to subdivide ab into smaller intervals.

In some cases, the size of V must be at least quadratic in the size of $\mathcal{G}$. This bound can be shown using the example of Figure 6.1 which consists of $m = |E|$ edges and $n = |S| = 2m + 2k$ vertices. The edges are parallel and very close to each other. The isolated vertices come in k

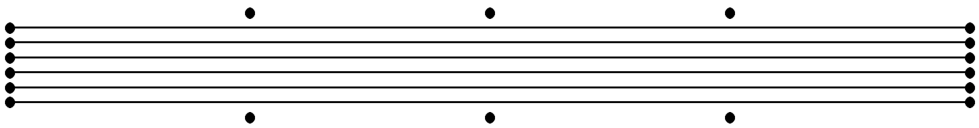

Figure 6.1: An $\Omega(mn)$ lower bound example on the number of vertices of a conforming Delaunay triangulation. In the example shown, $m = 6$, $k = 3$, and therefore $n = 2m + 2k = 18$.

pairs, with one vertex on each side of the group of edges. Provided the edges are sufficiently close to each other, and the vertices are sufficiently close to the edges, it will be necessary to place a point approximately between the two vertices of every pair on every edge. This proves that at least mk points need to be added to obtain a conforming Delaunay triangulation. The lower bound of $\Omega(mn)$ follows for $k = \Omega(n)$. For smaller k, the endpoints of half of the m edges can be used to play the role of the isolated vertices.

Prior Work. A common approach to produce a conforming Delaunay triangulation is to place sufficiently many points on the edges of the plane geometric graph so that each interval has a circle that avoids all other edges [Bois88, NaSr91, Olou91, Saal91, SaPe91]. This can always be achieved except maybe at places close to shared endpoints where sharp angles are formed. This special case is handled by placing points at the intersections of the edges with a sufficiently small circle drawn around the common endpoint. The method avoids the need for backtracking as no point placed on any edge harms any interval that already has such a circle. The price, however, is a possibly horrendous number of new points. Indeed, there is no function $f(n)$ that can bound the number of points although for every problem instance it is finite. In particular, the number of points added grows as the edges move closer to each other.

An Exponential Upper Bound. An upper bound that depends solely on n can be obtained as follows. Initially, set $V := S$ and consider all m edges as *unprotected*. Treat the edges of $\mathcal{G}$ in turn. At the time the ith edge is treated, it may consist of various protected and unprotected intervals. Place sufficiently many points on the unprotected intervals so that each new interval has a circle that does not enclose any point of the current set V. Each such circle may, however, intersect other edges. To prevent points from being placed inside the circle later in the process, we place points at the intersections between the circle and any unprotected interval of another edge. Declare each new interval as protected if it is enclosed by the circle and unprotected otherwise. See Figure 6.2.

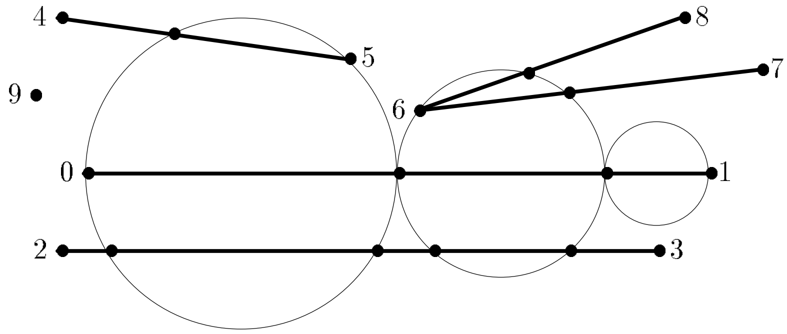

Figure 6.2: The original graph, $\mathcal{G}$, has vertices 0 through 9 and edges 01, 23, 45, 67 and 68. First, edge 01 is treated. In the process, two new points are added to 01. The resulting three circles intersect the other edges in seven points, which are also added. Each edge of $\mathcal{G}$ consists now of protected and unprotected intervals.

The number of points needed to treat the ith edge does not exceed the current size of V since it suffices to project the current set V orthogonally onto the ith edge. Similarly, the number of circles needed for the ith edge does not exceed the current size of V. Assume inductively that $|V| \leq n(2m+1)^{i-1}$ before the next step that treats the ith edge. The next step creates at most $n(2m+1)^{i-1}$ circles intersecting the remaining edges in at most $2m$ points each. The size of V thus increases to at most $n(2m+1)^{i-1} + n(2m+1)^{i-1}2m = n(2m+1)^i$. The total number of points at the end of the process is therefore at most $n(2m+1)^m$.

This method apparently produces far too many points. An improvement was found by Mehlhorn, Sharir, and Welzl. Their method combines the projection of points with a divide-and-conquer scheme and achieves a subexponential although not yet polynomial bound. The idea of protected and unprotected intervals turns out to be valuable in our effort to obtain a polynomial upper bound on the number of points.

6.2 The Upper Bound

Given a plane geometric graph $\mathcal{G} = (S, E)$, with $|S| = n$ and $|E| = m$, this section shows how to find $O(m^2 n)$ points so that each resulting interval has the empty disk property. As defined earlier, an interval is the portion of an edge between two contiguous points of V chosen on this edge. If no point of an edge belongs to V, then this edge itself is an interval.

6.2.1 The Global Idea

The point set V is constructed in two steps, the *blocking* and the *propagation phase.* Initially, V contains only the vertices of $\mathcal{G}$, that is, $V := S$.

The goal of the blocking phase is to find $O(n)$ pairwise disjoint disks that contain no points of S so that the union of their closures is connected and contains S. Each circle bounding such a disk is called a *blocking circle.* After finding these disks, we add the intersections between blocking circles and edges of $\mathcal{G}$ to the set V. In addition, we add the $O(n)$ points at which blocking circles touch each other. The new set V forms the vertex set W of a plane geometric graph $\mathcal{H}$ which conforms to $\mathcal{G}$. The edges of $\mathcal{H}$ are the intervals of $\mathcal{G}$ together with edges connecting contiguous points of V on blocking circles.

$\mathcal{H}$ has two types of edges. Each *protected* edge is enclosed by a blocking circle; its endpoints lie on the blocking circle. All other edges are *unprotected.* By construction, protected edges have the empty disk property with respect to the current set V. We will make sure that no points inside the blocking circles are later added to V so that this property persists with respect to all future sets V.

The unprotected edges are further subdivided into intervals in the propagation phase. For an edge or interval ab, we define the *minidisk* of ab, denoted by M_{ab}, as the smallest disk containing ab. In other words, M_{ab} is the disk bounded by the circle with ab as a diameter. If ab is unprotected and its minidisk contains a point $c \in V$ visible from every point of ab then a point c' subdividing ab into ac' and $c'b$ is added to V. This point c' will be chosen so that c lies outside $M_{ac'}$ and $M_{c'b}$.

6.2.2 The Details

<u>THE BLOCKING PHASE</u>. We show how to use a minimum spanning tree of S to construct $n-1$ disks $D_1, D_2, \ldots, D_{n-1}$ that satisfy the following properties:

(1) $D_i \cap S = \emptyset$ for all $1 \le i \le n-1$,

(2) $D_i \cap D_j = \emptyset$ for all $1 \le i < j \le n-1$,

(3) $D = \bigcup_{i=1}^{n-1}$ (closure of D_i) is connected, and

(4) $S \subseteq D$.

Recall that a minimum spanning tree of S, $mst(S)$, is a spanning tree of the complete graph $(S, \binom{S}{2})$ whose sum of edge lengths is a minimum (Section 2.4). An important property of $mst(S)$ is that the minidisk of every edge of $mst(S)$ is disjoint from S.

Label the vertices of $mst(S)$ from 0 through $n-1$ so that for every $0 \le j \le n-1$ the vertices $0, 1, \ldots, j$ induce a subtree of $mst(S)$. Define i_j so that $i_j j$ is an edge of this subtree. Notice that $i_j < j$ and that i_j is unique. The edges $i_j j$ are now used to define the disks D_j. The disk D_1 is the minidisk of edge 01. The disk D_j, for $2 \le j \le n-1$, is maximal so that

(i) its center lies on $i_j j$,

(ii) its bounding circle goes through j, and

(iii) it is disjoint from disks D_1 through D_{j-1} constructed earlier.

Clearly, $D_j \subseteq M_{i_j j}$. This implies property (1). Properties (2), (3), and (4) follow from the construction.

Let S' be the set of points where the blocking circles intersect the edges of $\mathcal{G}$, and let S'' be the set of points (not in S') where the blocking circles touch each other. As described above, the points in S' and S'' are added to V. Let W be the new set V, and let $\mathcal{H} = (W, F)$ be a plane geometric graph with $F = F' \cup F''$ defined as follows. The set F' contains all intervals on edges of $\mathcal{G}$. Remember that by construction all points of W lie on the $n-1$ blocking circles.

Consider a disk D_i bounded by a blocking circle C_i, and let $p_0, p_1, \ldots, p_{k-1}, p_k = p_0$ be the points of W that lie on C_i in this sequence. These points define a convex k-gon with edges $p_\ell p_{\ell+1}$, $0 \leq \ell \leq k-1$, termed *walls*. Some of these walls may be intervals on edges of $\mathcal{G}$ and therefore belong to F'. In any case, F'' is the collection of all walls. This completes the definition of $\mathcal{H}$ which conforms to $\mathcal{G}$; see Figure 6.3.

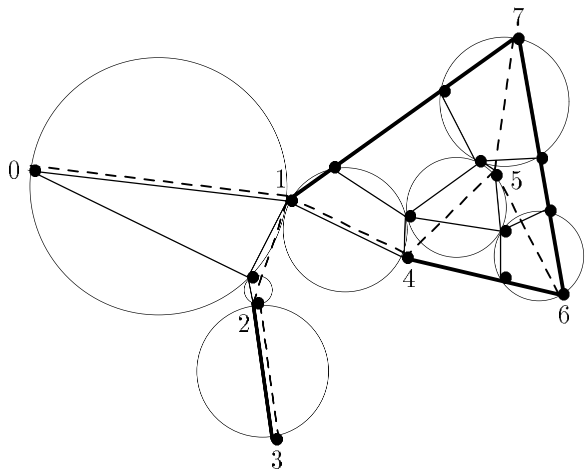

Figure 6.3: The original graph, $\mathcal{G}$, has vertices 0 through 7 and edges 17, 23, 46 and 67. The edges of the minimum spanning tree, 01, 12, 23, 14, 45, 56 and 57, are indicated by broken lines. Each edge of the tree corresponds to a blocking circle. Each blocking circle encloses protected edges of $\mathcal{H}$.

Note that each wall is protected by a blocking circle. The collection of walls defines another plane geometric graph, $\mathcal{I} = (W, F'')$. Clearly, $\mathcal{I}$ is a subgraph of $\mathcal{H}$. Note that all faces of $\mathcal{H}$, excluding the unbounded face, are simply connected because $\mathcal{H}$ is connected. For convenience, we shall adopt the topology of the sphere in our discussion so that *all* faces of $\mathcal{H}$ are simply connected. Similarly, all faces of $\mathcal{I}$ are simply connected because $\mathcal{I}$ is connected.

<u>**The Propagation Phase**</u>. The unprotected edges of $\mathcal{H}$ are further subdivided into intervals during a non-deterministic process. Initially, every unprotected edge is also an unprotected interval. Consider an unprotected interval ab and its minidisk M_{ab}. Call a point $c \in V$ *visible* from ab if the interior of abc is disjoint from all edges of $\mathcal{H}$. Suppose a point $c \in V$ visible from ab is contained in M_{ab}. We add c' to V, where c' is the orthogonal projection of c onto ab, and thus subdivide ab into ac' and $c'b$. Repeat this step until there is no unprotected interval ab with such a point c. This completes the description of how the point set V is constructed.

6.2.3 The Analysis

This subsection shows that the eventual size of V is $O(m^2n)$. The blocking phase adds at most $(2m-1)(n-1)$ intersection points between edges and circles and fewer than $3n$ points where circles touch each other. The latter bound follows from the planarity of the intersection graph of the blocking circles.

We now focus on proving that each point created in the blocking phase gives rise to at most $O(m)$ points in the propagation phase. We begin by proving a few properties of the propagation phrase. Let ab be an unprotected interval at some point in time during the process, and let $c \in V$ lie in M_{ab}. Then the orthogonal projection c' of c onto the line through ab lies strictly between a and b, i.e., $c' \in ab$. Furthermore, $c \notin M_{ac'}$ and also $c \notin M_{c'b}$.

Assume now that $c \neq x, y$ is a point on some edge xy of $\mathcal{G}$. All points in $V - (S \cup S'')$ are of this form. Then we have the following fact.

Lemma 6.2 There are at most two intervals, ab and $a'b'$, so that c is visible from both and contained in their minidisks. Furthermore, ab and $a'b'$ lie on different sides of the line through xy.

Proof. If $c \in M_{ab}$ then $\angle acb > \frac{\pi}{2}$, and if c is visible from ab then a and b lie on the same side of the line through xy. The same is true for $a'b'$. So if ab and $a'b'$ lie on the same side of the line then one endpoint of $a'b'$ must lie between a and b as seen from c. This contradicts the assumption that c is visible from ab and from $a'b'$. $\square$

Next, let $c \in M_{ab}$ be visible from ab, and c' be the orthogonal projection of c onto ab. Then we have the following fact.

Lemma 6.3 There is no interval $a''b''$ on the same side of the line through ab as c so that c' is visible from $a''b''$ and $c' \in M_{a''b''}$.

Proof. Assume such an $a''b''$ exists. Then $\angle a''c'b'' > \frac{\pi}{2}$ which implies that c lies between a'' and b'' as seen from c'. This either contradicts that c is visible from ab or that c' is visible from $a''b''$. $\square$

Assume now that $c \in V$ does not lie on an edge of $\mathcal{G}$, so $c \in S \cup S''$. Similar to Lemma 6.2, we have the following fact as there can be at most three angles larger than $\frac{\pi}{2}$ packed around c.

Lemma 6.4 There are at most three intervals ab so that c is visible from ab and $c \in M_{ab}$.

<u>A LOCALITY PROPERTY</u>. Notice that the propagation phase takes care only of local constraints. In other words, it considers only visible point-interval pairs c, ab. Although the minidisk of ab can contain other points of V, it is indeed justified to ignore such points, as we will see shortly. Let ab be an unprotected interval. We call the minidisk M_{ab} *locally empty* if it contains no point of V that is visible from ab. Furthermore, M_{ab} is *empty* if it contains no point of V at all.

To prepare for the next lemma, we consider an interval ab and a point $c \in M_{ab}$. If c is not visible from ab then there are intervals st that intersect abc. We say that st *separates* c from ab if both endpoints, s and t, lie outside M_{ab}. Otherwise, st *hinders* the visibility between c and ab but it does not separate. Let $E_{c,ab}$ be the set of intervals that separate c and ab, and define $F_{c,ab}$ as the set of (non-separating) intervals that hinder the visibility between c and ab. It is interesting to observe that $F_{c,ab} = \emptyset$ or there is another point d of V in M_{ab} with $E_{d,ab} \subseteq E_{c,ab}$ and $F_{d,ab} = \emptyset$. To see this, choose a point x in the interior of abc so that the interior of abx does not intersect any edge of $\mathcal{H}$. Move x continuously and straight towards c until either a side of abx hits a vertex d or x hits a non-separating interval uv of $\mathcal{H}$. In the second case at least one of the two endpoints, say u, lies in M_{ab}. Slide x on uv towards u until either a side of abx hits another vertex d or x reaches u (then $d = u$), whichever happens first. The path of x crosses only intervals that separate c and ab, therefore $E_{d,ab} \subseteq E_{c,ab}$.

Lemma 6.5 If the minidisks of all unprotected intervals are locally empty then they are all empty.

Proof. Suppose the claim is false. Then, there is an unprotected interval ab whose minidisk M_{ab} contains a point $c \in V$. As argued in the preceding paragraph, we can make the extremal assumption that ab and c are chosen so that the number of separating intervals $E_{c,ab}$ is a global minimum and $F_{c,ab} = \emptyset$. Let st be an interval that separates ab from c. Note that st is not protected because every circle through s and t encloses at least one of a, b and c. But if st is an unprotected interval, we have $c \in M_{st}$ because s and t are outside M_{ab}. Furthermore, st cuts M_{ab} into two pieces, and the piece that contains c is properly contained in M_{st}. Therefore $E_{c,st} \subseteq E_{c,ab}$ and we have proper containment because $st \in E_{c,ab}$ does not belong to $E_{c,st}$. This either contradicts the extremal assumption or that st is locally empty. $\square$

PROPAGATION SEQUENCES. We are now ready to analyze the number of points created in the propagation phase. Our particular goal is to show that each point c created in the blocking phase generates at most $3m$ points in the propagation phase. We say that a point c *generates* another point d if there is a sequence $c = c_0, c_1, \ldots, c_k = d$ so that c_{i+1} is created as the orthogonal projection of c_i onto some interval during the propagation phase, for $0 \leq i \leq k-1$. The sequence $c_0, c_1, \ldots, c_k$ is called a *propagation sequence*. It is *non-trivial* if $k \geq 1$, and it is *maximal* if c_0 is created in the blocking phase and c_k generates no further point. Note that all c_i, $i \geq 1$, of a maximal propagation sequence lie on unprotected edges of $\mathcal{H}$, and these edges are contained in edges of $\mathcal{G}$.

Every point d created in the propagation phase gives rise to at most one point d'. To see this, we first note that such a point d lies on an edge xy of $\mathcal{G}$. Now Lemma 6.2 implies that d is visible from at most two intervals whose minidisks contain d. By Lemma 6.3 and because d itself is generated by an orthogonal projection, d generates another point d' on at most one of these two intervals. Together with Lemma 6.4, this implies that a point of V constructed before the propagation phase gives rise to at most three non-trivial maximal propagation sequences. In fact, there are at most two such sequences per point not in $S \cup S''$. To establish that the length of a maximal propagation sequence is at most m, it suffices to prove the following.

Lemma 6.6 No propagation sequence can have two or more points on the same edge of $\mathcal{G}$.

Proof. Suppose the claim is false. Consider a minimal propagation sequence, $c = c_0, c_1, \ldots, c_k = d$, so that c and d lie on the same edge xy of $\mathcal{G}$. Consider the polygon, P, whose boundary consists of the line segments cd and $c_i c_{i+1}$, for $0 \leq i \leq k-1$. Recall that walls are protected, so no points are projected on or across them. Thus, P lies completely within a face of $\mathcal{I}$. Since all vertices of $\mathcal{G}$ are also vertices of $\mathcal{I}$, and because all faces of $\mathcal{I}$ are simply connected, there can be no vertex of $\mathcal{G}$ inside P. Thus, each edge of $\mathcal{G}$ that intersects P has its endpoints outside P. It thus intersects the boundary of P in at least two points. Since $c_0, c_1, \ldots, c_k$ is minimal, it follows that $k = 2$ and that c_0 and c_2 lie on the same side of the edge xy of $\mathcal{G}$ that contains c_1. But this contradicts Lemma 6.3. $\square$

As mentioned earlier, $|S \cup S''| < 4n$. Each point $c \in S \cup S''$ gives rise to at most three maximal propagation sequences of length at most $m + 1$ each. The point c itself is the only point of these sequences that does not necessarily lie on an edge of $\mathcal{G}$. The number of other points created during the blocking phase is less than $2mn$. Each such point gives rise to at most two maximal propagation sequences of length at most m each. The total number of points after the propagation phase is thus less than

$$4n(3m + 1) + 2mn(2m - 1) = 4m^2n + 10mn + 4n.$$

This proves the main result of this section.

Theorem 6.7 Let $\mathcal{G} = (S, E)$ be a plane geometric graph with $|S| = n$ and $|E| = m \geq 1$. There exists a point set V of size $|V| = \mathrm{O}(m^2 n)$ so that its Delaunay triangulation has a completion that conforms to $\mathcal{G}$.

6.3 Implementing the Proof

The proof of the $\mathrm{O}(m^2 n)$ upper bound presented in Section 6.2 is constructive and can be translated into an algorithm without much effort. The only demanding step is the implementation of the propagation phase. In order to keep the time-complexity roughly within the same order of magnitude as the number of points added, we need to project the points in a sequence that is computationally inexpensive. We will assume that point coordinates can be stored in constant amount of storage and that basic geometric operations, such as intersecting a circle with an edge and projecting a point onto a line, can be carried out in constant amount of time. As usual, plane geometric graphs are stored using the quad-edge data structure of Guibas and Stolfi (Section 1.2).

6.3.1 The Blocking Phase

A minimum spanning tree of a set of n points in the plane can be computed in time $O(n \log n)$. This requires the construction of the Delaunay triangulation of the points and running a standard minimum spanning tree algorithm on the graph of this triangulation; see, for example, [Edel87, Section 13.2.5]. Alternatively, a minimum spanning tree can be obtained in time $O(n^2)$ directly from the complete graph of the points. The slower method is certainly easier to implement.

After computing the tree, we need to find the disks $D_1, D_2, \ldots, D_{n-1}$ that satisfy properties (1) through (4). Most straightforwardly, these disks can be constructed one by one as explained in Section 6.2. For each j, the largest disk D_j needs to be found so that its center lies on $i_j j$, j lies on the bounding circle of D_j, and D_j avoids all D_i for $i < j$. This can be done in time $O(j)$. The total amount of time for this step is thus $O(n^2)$. The plane geometric graphs $\mathcal{H}$ and $\mathcal{I}$ can be computed by intersecting the bounding circles of the D_j with each other and with the edges of $\mathcal{G}$. The resulting $O(mn)$ intersection points can be computed and sorted along circles and edges in time $O(mn \log n)$.

6.3.2 The Propagation Phase

Recall that $\mathcal{I}$ is a subgraph of $\mathcal{H}$ and contains none of its unprotected edges. A point $c \in V$ is projected onto an interval ab only if ab is unprotected. Thus, projections happen only within faces of $\mathcal{I}$. We can thus restrict our attention to a single face of $\mathcal{I}$. As mentioned earlier, $\mathcal{I}$ is connected and therefore its faces are simply connected. Each face of $\mathcal{I}$ is further subdivided into *regions* (of $\mathcal{H}$) by unprotected edges of $\mathcal{H}$. Each region is also simply connected, and can be treated computationally as bounded by a simple polygon defined by oriented edges that have the region on their left (Section 1.2).

A TREE OF REGIONS. Consider a face f of $\mathcal{I}$. Let $\mathcal{R}_f$ be the graph whose nodes are the regions of f, and whose arcs connect regions that share unprotected edges of $\mathcal{H}$. Each subdividing unprotected edge has both endpoints on the boundary of f, which implies that $\mathcal{R}_f$ is a free tree. It will be convenient to fix an arbitrary node as its root and thus impose a parent-child relation on adjacent node pairs. Points will be projected onto unprotected edges in three stages. The first stage computes an initial set of projections that avoids difficult situations in the second stage. The second stage computes and sorts segments along the boundary of each region. The last stage consists of a post-order and a pre-order traversal of $\mathcal{R}_f$.

Consider two adjacent nodes ν and ξ of $\mathcal{R}_f$, and let ab be their shared unprotected edge. Points on the boundary of ν that are projected onto ab are said to be *exported* from ν to ξ. Symmetrically, we say they are *imported* by ξ. The points projected onto ab are stored in two separate sorted lists, $L_{\nu\xi}$ and $L_{\xi\nu}$, one for each side of ab. The complete list of points exported from ν to ξ, $L_{\nu\xi}$, can be computed only after all import lists $L_{\kappa\nu}$, for $\kappa \neq \xi$ adjacent to ν, are available. Note that the import list from ξ, $L_{\xi\nu}$, is not necessary for computing $L_{\nu\xi}$ because a propagation sequence follows only one direction of a path in $\mathcal{R}_f$ (Lemma 6.6).

<u>**Stage 1**</u>: *subdividing unprotected edges.* A vertex v_j of ν is *reflex* if the angle inside ν exceeds π, i.e., vertices v_i, v_j, v_k form a right turn where $\vec{v_i v_j}$ and $\vec{v_j v_k}$ are the oriented edges on the boundary of ν. The first stage projects every reflex vertex c onto all unprotected edges ab for which there exists a portion $a'b' \subseteq ab$ so that c is visible from $a'b'$ and $c \in M_{a'b'}$. Since $\angle a'cb' > \frac{\pi}{2}$ there can be at most three such edges ab. As a precaution, we do not require that c be visible from ab. This way c does not need to be reconsidered after ab gets subdivided. Although such projections are not prescribed by the proof in Section 6.2, they neither invalidate the correctness nor the analysis of the construction. Each reflex vertex is necessarily a vertex in $S \cup S''$, so there are fewer than $4n$ of them. Since each vertex is projected at most three times, we thus increase the number of unprotected edges by less than $12n$.

Here is how we find the at most three unprotected edges for a reflex vertex c. The parts of the boundary of ν visible from c can be computed in a single walk along the boundary of ν; see, for example, [ElAv81, Lee83, JoSi87]. The amount of time needed for the walk is proportional to the number of edges. Select the at most three edges that have connected portions visible from c along an angle exceeding $\frac{\pi}{2}$. Project c orthogonally onto these at most three edges. Each projection subdivides an unprotected edge into two such edges.

The rest of the algorithm uses the subdivision of $\mathcal{H}$ produced in stage 1. It will therefore be convenient to call the elements of this subdivision vertices and edges. After the completion of stage 1, no reflex vertex visible from an unprotected edge ab lies inside the minidisk M_{ab}. It thus follows that if a point c lies in M_{ab} and is visible from a point on ab then it is also visible from ab.

<u>Stage 2</u>: *computing boundary segments.* This stage is a preprocessing step that speeds up computations in stage 3. It prepares the boundary of ν in such a way that points can be projected onto various unprotected edges in a single walk along the boundary of ν. To this end, we associate pieces of the boundary, called *segments*, with the unprotected edges of ν. A *segment* for an unprotected edge ab is a maximal connected piece of ν's boundary so that every point x of the segment is visible from ab and contained in M_{ab}. Note that segments do not include their endpoints. Since ν is simply connected, a point x is visible from ab iff it is visible from a and also from b. The segments of ab are constructed as follows.

1. Find the part of ν's boundary visible from a. As mentioned above, this can be done in a single walk along the boundary.
2. Find the part of ν's boundary that is also visible from b. Again a single walk suffices.
3. Intersect the identified boundary pieces with M_{ab}.

Define the *rank* of ν, $r(\nu)$, equal to the number of unprotected edges of ν, and let $|\nu|$ be the total number of edges of ν. If an edge bounds ν on both sides, that is, the edge belongs to the interior of the closure of ν, then it is counted twice. By construction, this can be the case only for protected edges. After carrying out 1 through 3 for each unprotected edge of ν, we obtain a collection of segments. Because of Lemma 6.2, the segments along the boundary of ν are pairwise disjoint. In other words, the segments form a sequence, and they can be sorted in time proportional to $|\nu|$ plus their number. This is because along an edge of ν the segments are ordered consistently with the order of their corresponding unprotected edges. We will see later that the number of segments is less than $2\,r(\nu) + |\nu|$.

<u>Stage 3</u>: *traversing the tree.* In a post-order traversal, the children κ of a node ν are visited before ν. Visiting a node ν in this case means computing the export list to its parent ξ. Notice that because of stage 1, ν and ξ may share several unprotected edges. Still, their union is the original unprotected edge of ν and ξ, and $L_{\nu\xi}$ can be obtained by concatenating the lists obtained by projecting points onto these unprotected edges. We can assume that at the time the export list of ν to ξ is computed, all import lists $L_{\kappa\nu}$ are available.

List $L_{\nu\xi}$ is constructed in a single walk along the boundary of ν. Whenever a segment that belongs to an unprotected edge shared by ν and ξ is encountered, the points on this segment are projected orthogonally onto the unprotected edge. These points can be vertices of ν or points in import lists of ν. The result is the list $L_{\nu\xi}$. It is automatically sorted if we process the points in their order along the boundary of ν.

After the post-order traversal of $\mathcal{R}_f$, all child-to-parent lists are complete. In order to compute the export lists of a node ν to its children κ, we need to first construct the import list from its parent, $L_{\xi\nu}$. This is done in a pre-order traversal of $\mathcal{R}_f$. A node is visited before its children, and visiting a node ν now means computing all export lists $L_{\nu\kappa}$. At the time we compute these lists, all import lists are complete and stored with their unprotected edges. In a final walk along the boundary of ν, we project vertices onto the appropriate unprotected edges, as before.

6.3.3 The Time Complexity

The analysis of the above algorithm requires some topological and combinatorial results about regions. We begin with a combinatorial lemma. Let $e_1, e_2, \ldots, e_k$ be a sequence of k not necessarily distinct symbols. It is a *$DS_2(n)$-sequence* if only n of the symbols e_i are different, $e_i \neq e_{i+1}$ for $1 \leq i \leq k-1$, and there are no four indices $1 \leq i_1 < i_2 < i_3 < i_4 \leq k$ so that $e_{i_1} = e_{i_3} \neq e_{i_2} = e_{i_4}$ [DaSc65]. The following is a well-known fact about the *length* k of the sequence.

Lemma 6.8 The length of any $DS_2(n)$-sequence is at most $2n-1$.

Proof. Suppose z is the symbol that is introduced last when a $DS_2(n)$-sequence is read from left to right. From the last part of the definition, we deduce that z occurs only once. We can thus obtain a $DS_2(n-1)$-sequence by either deleting z or z and its adjacent symbol. The latter case happens only when z is sandwiched between a same symbol. Clearly, induction on n completes the proof since $DS_2(1) = 1$. $\square$

We use such sequences to bound the number of segments in a region ν.

Lemma 6.9 The number of segments of ν is less than $2\,r(\nu) + |\nu|$.

Proof. Consider the ordered sequence of segments. Replace each segment by the name of the corresponding unprotected edge. The resulting sequence contains no scattered subsequence of the form

$$\ldots ab \ldots cd \ldots ab \ldots cd \ldots,$$

because otherwise the bounding circles of M_{ab} and M_{cd} would intersect at four or more points. So if we compress repetitions we get a $DS_2(r(\nu))$-sequence. If two consecutive symbols (unprotected edges) are the same then there must be a vertex of ν separating them. This implies that the total number of segments exceeds the length of the $DS_2(r(\nu))$-sequence by at most $|\nu|$. $\square$

The total number of unprotected edges before the propagation phase is at most $\mathrm{O}(mn)$, and it is fairly easy to see that this bound is tight. It is plausible that a single region can have only substantially fewer unprotected edges. We now prove a more general result that implies a single region indeed cannot exceed $\mathrm{O}(m+n)$ unprotected edges. For a region ν, we define its *excess* $e(\nu) = \max\{0, r(\nu) - 4m\}$. The *total excess* is the sum of the $e(\nu)$ over all regions ν of $\mathcal{H}$. We prove an upper bound on the total excess which is sufficient for our purposes but certainly not tight.

Lemma 6.10 The total excess is less than $36n$.

Proof. To help the discussion we replace each edge of $\mathcal{G}$ by a pair of directed edges, which we call *di-edges*. A di-edge *pq contributes* an edge to a region ν if it contains the edge and along this edge ν lies to the left of *pq*. Consider the sequence of unprotected edges in ν's boundary, and replace each such edge by the name of the contributing di-edge of $\mathcal{G}$. This results in a sequence with at most $2m$ different symbols. A straightforward topological argument shows that there is no scattered subsequence of the form

$$\ldots pq \ldots st \ldots pq \ldots st \ldots$$

If we ignore repetitions we have a $DS_2(2m)$-sequence, which implies that the length of the sequence without repetitions is less than $4m$. Anything exceeding this number is counted by $e(\nu)$.

Let ab and cd be two consecutive unprotected edges contributed by the same di-edge, pq. Then (i) $b = c$, or (ii) bc is a protected edge of ν, or (iii) there are two or more protected edges between b and c. In case (i) we can charge the projection in stage 1 for the repetition, and there are fewer than $12n$ of them. Each projected point is counted twice, once for each side, so we have fewer than $24n$ repetitions of type (i) in total. In case (iii) we can charge the vertex common to the first two protected edges after b for the repetition. This vertex must be in $S \cup S''$. We have $|S \cup S''| < 4n$, and each such vertex is charged at most twice because it can lie on at most two blocking circles. This implies that there are fewer than $8n$ repetitions of type (iii). In case (ii), bc is a protected edge contributed by pq. We argue in the following that the total number of such edges, over all regions and di-edges, is less than $4n$. This will imply the claim.

Since bc is protected, its endpoints lie on a blocking circle C_j bounding D_j. Furthermore, since bc belongs to a region with at least one unprotected edge, it must be a wall. Consider D_j and the edges of $\mathcal{H}$ that lie on edges of $\mathcal{G}$ and decompose D_j. Each such edge has both endpoints on C_j. It follows that the dual graph of the decomposition is a free tree. The nodes of the tree are the regions of the decomposition, and the arcs correspond to the edges that decompose D_j. The edge bc corresponds to an arc incident to a leaf of the dual graph. We can bound the number of repetitions of type (ii) by bounding the total number of leaves of the $n - 1$ dual graphs defined for the blocking circles.

To count the total number of leaves, we assume $\mathcal{G}$ is a triangulation. If not, it can be converted into one by adding fewer than $3n - m$ edges; adding these edges can only increase the count. The advantage of a triangulation is that now each interior node of a dual graph has degree 2 or 3. Furthermore, the number of leaves of a dual graph is 2 plus the number of degree-3 nodes. Each degree-3 node of the dual graph for D_j corresponds to a triangle of $\mathcal{G}$ each of whose three sides intersects C_j. A triangle can intersect at most one blocking circle in this manner. Thus, the total number of degree-3 nodes is at most the number of triangles, i.e., $2n - 5$. This number plus twice the number of blocking circles is less than $4n$, as claimed. $\square$

A region cannot have more than $4n$ vertices shared by adjacent protected edges, because each such vertex is a vertex in $S \cup S''$. Each such vertex is encountered at most twice, which implies $|\nu| \leq 2\, r(\nu) + 8n$. Lemma 6.10 thus implies a bound on the number of edges of a region.

Lemma 6.11 For a region ν of $\mathcal{H}$, $|\nu| = \mathrm{O}(m+n)$.

We are now ready to derive the time-complexity of the algorithm. Stages 1 and 2 require at most $\mathrm{O}(|\nu|\, r(\nu))$ time per region ν. By definition, $r(\nu) \leq 4m + e(\nu)$, so the time is bounded by a constant times

$$\sum |\nu|\, m + \sum |\nu|\, e(\nu),$$

where the sums are over all regions of $\mathcal{H}$. The first sum is $\mathrm{O}(m^2 n)$ because $\sum |\nu| = \mathrm{O}(mn)$ for $\mathcal{H}$ has only $\mathrm{O}(mn)$ edges and stage 1 adds only $\mathrm{O}(n)$ to this number. The second sum is $\mathrm{O}(n^2)$ because $\sum |\nu|\, e(\nu) = \mathrm{O}((m+n) \sum e(\nu))$ by Lemma 6.11 and $\sum e(\nu) < 36n$ by Lemma 6.10. This implies that $\mathrm{O}(m^2 n + n^2)$ is an upper bound for the time spend in the first two stages of the algorithm. After that, $\mathrm{O}(m^2 n)$ time suffices to compute all import and export lists in stage 3. This implies the main result of this section.

Theorem 6.12 Let $\mathcal{G} = (S, E)$ be a plane geometric graph with $|S| = n$ and $|E| = m \geq 1$. A point set V of size $\mathrm{O}(m^2 n)$ that admits a completion of its Delaunay triangulation conforming to $\mathcal{G}$ can be computed in time $\mathrm{O}(m^2 n + n^2)$.

6.4 Discussion

The main result of this chapter is the existence of $\mathrm{O}(m^2 n)$ points that admit a completion of their Delaunay triangulation conforming to a plane geometric graph with n vertices and $m \geq 1$ edges. This result is superficially similar to the triangulation results of [BDE92, BEG90, BeEp91, MeSo92]. The best lower bound for the number of points necessary is $\Omega(mn)$, and its proof is fairly straightforward. It would be interesting to close the gap between the two bounds.

The $\mathrm{O}(m^2 n)$ points can be constructed in time $\mathrm{O}(m^2 n + n^2)$, provided infinite precision arithmetic in constant time is assumed. This assumption is unrealistic because the number of bits necessary to accurately represent a point increases at each projection along a propagation sequence. It remains open whether the points can be constructed within the same time-bound without this assumption.

Acknowledgments

This is joint work of Herbert Edelsbrunner and myself [EdTa92]. It benefits from communications with a few people. We are grateful to Sang-Ho Lee (Ewha Womans University, Korea) for helpful discussions during his visit at the University of Illinois, and Kurt Mehlhorn (Max-Planck Institut, Germany), Micha Sharir (Tel-Aviv University, Israel), and Emo Welzl (Freie Universität Berlin, Germany) for communicating their improvement of the exponential bound described in Section 6.1. These results eventually lead to the key idea of the upper bound proof, that is, the concept of walls.

Chapter 7

Conclusion

Many words have been said in the literature about triangulation for it is, no doubt, an important decomposition method with various applications in engineering and science. This thesis focuses on the complexity of constructing optimal triangulations. This theme has not been very popular in the literature until very recently. Because of a lack of algorithms, most previous work on constructing optimal triangulations relies on simple heuristics. This thesis substantially advances our knowledge in this respect and supplements the meager pool of algorithmic methods known to be effective in constructing triangulations.

Now we can compute min-max angle, max-min height, min-max slope, min-max eccentricity, min-max length, and conforming Delaunay triangulations in low order polynomial time. Besides formulating new algorithmic and proving them correct, we also address implementation issues. All algorithms discovered are fairly simple and have interesting visual effects when animated. Roughly speaking, an edge-insertion step (Chapters 3 and 4) looks as if it is opening and then closing a hand-fan. The Subgraph approach (Chapter 5) grows triangles inside polygons. The "wall" approach (Chapter 6) marches up and down faces. We observed the behavior of an edge-insertion step from the implementation done by Roman Waupotitsch [EdWa92]; the other two approaches are not yet implemented.

This thesis is certainly not the final word in the area. As discussed in Section 1.3, many questions remain open. Besides two-dimensional cases, there are many three- and higher-dimensional problems; see, for example, [BeEp92, Dey91, Edel90, EdSh92, MiVa92, Raja91]. Just by going up one dimension to three, we reach another *dimension of difficulty.* Simple questions about the complexity of triangulating a convex polytope optimally remain open. Moreover, polytopes may not necessarily have triangulations [Schö28], and the corresponding decision problem is NP-complete and thus unlikely to have a polynomial time algorithm [RuSe92]. This explains some difficulties on extending the results in this thesis to higher dimensions. We leave with many words yet to be said about triangulations ......

Bibliography

[ACNS82] M. Ajtai, V. Chvátal, M. M. Newborn and E. Szemerédi. Crossing-free subgraphs. *Ann. Discrete Math.* **12** (1982), 9–12.

[AHU74] A. V. Aho, J. E. Hopcroft and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addision-Wesley, Reading, MA, 1974.

[Akim84] H. Akima. On estimating partial derivatives for bivariate interpolation of scattered data. *Rocky Mountain J. Math.* **14** (1984), 41–52.

[Aure91] F. Aurenhammer. Voronoi diagrams—A survey of a fundamental geometric data structure. *ACM Comput. Surveys* **23** (1991), 345–405.

[BaAz76] I. Babuška and A. K. Aziz. On the angle condition in the finite element method. *SIAM J. Numer. Anal.* **13** (1976), 214–226.

[BaLi84] R. E. Barnhill and F. F. Little. Three- and four-dimensional surfaces. *Rocky Mountain J. Math.* **14** (1984), 77–102.

[Barn77] R. E. Barnhill. Representation and approximation of surfaces. *Math. Software III*, J. R. Rice, ed., Academic Press, 1977, 69–120.

[BaSu90] I. Babuška and M. Suri. The *p*- and *h-p* versions of the finite element method, an overview. *Comput. Methods Appl. Mech. Engrg.* **80** (1990), 5–26.

[BDE92] M. Bern, D. Dobkin and D. Eppstein. Triangulating polygons without large angles. *In* "Proc. 8th Ann. Sympos. Comput. Geom., 1992", 222–231.

[BEEMT92] M. Bern, H. Edelsbrunner, D. Eppstein, S. Mitchell and T. S. Tan. Edge insertion for optimal triangulations. *In* "Proc. 1st Latin American Sympos. Theoretical Inform., 1992", *Lecture Notes in Comput. Sci.* **583** (1992), 46–60. Also to appear in *Discrete Comp. Geom.*

[BEG90] M. Bern, D. Eppstein and J. Gilbert. Provably good mesh generation. *In* "Proc. 31st Ann. IEEE Sympos. Found. Comput. Sci., 1990", 231–241.

[BeEp91] M. Bern and D. Eppstein. Polynomial-size nonobtuse triangulation of polygons. *In* "Proc. 7th Ann. Sympos. Comput. Geom., 1991", 342–350.

[BeEp92] M. Bern and D. Eppstein. Mesh generation and optimal triangulation. To appear in *Computing in Euclidean Geometry*, D. Z. Du and F. K. Hwang, eds., World Scientific, Singapore, 1992.

[BFL88] J. D. Boissonnat, O. D. Faugeras and E. Le Bras-Mehlman. Representing stereo data with the Delaunay triangulation. *In* "Proc. Internat. Conf. Robotics and Automation, 1988", 24–29.

[Bois88] J. D. Boissonnat. Shape reconstruction from planar cross sections. *Comput. Vision, Graphics and Image Process.* **44** (1988), 1–29.

[BoMu76] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications.* Maxmillan Press, Hong Kong, 1976.

[Brow79] K. Q. Brown. Voronoi diagrams from convex hulls. *Inform. Process. Lett.* **9** (1979), 223–228.

[BrZl70] J. Bramble and M. Zlámal. Triangular elments in the finite element method. *Math. Comp.* **24** (1970), 809–820.

[Cave74] J. Cavendish. Automatic triangulation of arbitrary planar domains for the finite element method. *Internat. J. Numer. Methods Engrg.* **8** (1974), 679–696.

[Chew89] L. P. Chew. Constrained Delaunay triangulation. *Algorithmica* **4** (1989), 97–108.

[CLR90] T. H. Cormen, C. E. Leiserson and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, MA, 1990.

[D'AS89] E. F. D'Azevedo and R. B. Simpson. On optimal interpolation triangle incidences. *SIAM J. Sci. Statist. Comput.* **10** (1989), 1063–1075.

[DaSc65] H. Davenport and A. Schinzel. A combinatorial problem connected with differential equations. *Amer. J. Math.* **87** (1965), 684–694.

[Dela34] B. Delaunay. Sur la sphère vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* **7** (1934), 793–800.

[Dey91] T. K. Dey. Decompositions of polyhedra in three dimensions. Techn. Rep. CSD-TR-91-056, Ph.D. Thesis, Comput. Sci. Dept., Purdue Univ., IN, 1991.

[DLR90a] N. Dyn, D. Levin and S. Rippa. Algorithms for the construction of data dependent triangulations. *Algorithms for Approximation II*, J. C. Mason and M. G. Cox, eds., Chapman and Hall, London, 1990, 185–192.

[DLR90b] N. Dyn, D. Levin and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA J. Numer. Anal.* **10** (1990), 137–154.

[Edel87] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Heidelberg, Germany, 1987.

[Edel90] H. Edelsbrunner. Spatial triangulations with dihedral angle conditions. *In* "Proc. Internat. Workshop on Discrete Algorithms and Complexity, 1989", 83–89.

[Edel92] H. Edelsbrunner. Geometric Algorithms. To appear in *Handbook on Convex Geometry*, P. Gruber and J. Wills, eds.

[EdMü90] H. Edelsbrunner and E. P. Mücke. Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms. *ACM Trans. Graphics* **9** (1990), 66–104.

[EdSh92] H. Edelsbrunner and N. R. Shah. Incremental topological flipping works for regular triangulations. *In* "Proc. 8th Ann. Sympos. Comput. Geom., 1992", 43–52.

[EdTa91] H. Edelsbrunner and T. S. Tan. A quadratic time algorithm for the minmax length triangulation. *In* "Proc. 32nd Ann. IEEE Sympos. Found. Comput. Sci., 1991", 414–423. Also to appear in *SIAM J. Comput.*

[EdTa92] H. Edelsbrunner and T. S. Tan. An upper bound for the conforming Delaunay triangulation. *In* "Proc. 8th Ann. Sympos. Comput. Geom., 1992", 53–62. Also to appear in *Discrete Comp. Geom.*

[EdWa92] H. Edelsbrunner and R. Waupotitsch. Optimal Two-dimensional triangulations. *Animation of Geometric Algorithms: A Video Review*, M. Brown and J. Hershberger, eds., Techn. Rep. 87a, Systems Research Center, DEC, Palo Alto, CA, 1992, 7–8.

[ElAv81] H. ElGindy and D. Avis. A linear algorithm for computing the visibility polygon from a point. *J. Algorithms* **2** (1981), 186–197.

[ETW90] H. Edelsbrunner, T. S. Tan and R. Waupotitsch. An $O(n^2 \log n)$ time algorithm for the minmax angle triangulation. *In* "Proc. 6th Ann. Sympos. Comput. Geom., 1990", 44–52. Also, *SIAM J. Sci. Statist. Comput.* **13** (1992), 994–1008.

[Fort87] S. Fortune. A sweepline algorithm for Voronoi diagrams. *Algorithmica* **2** (1987), 153–174.

[FrFi91] W. H. Frey and D. A. Field. Mesh relaxation: a new technique for improving triangulations. *Internat. J. Numer. Methods Engrg.* **31** (1991), 1121–1133.

[FrSc87] R. Franke and L. L. Schumaker. A bibliography of multivariate approximation. *Topics in Multivariate Approximation*, C. K. Chui, L. L. Schumaker and F. I. Utreras, eds., Academic Press, 1987, 275–335.

[GaJo79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, NY, 1979.

[GaSo69] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology* **18** (1969), 259–278.

[GCR77] C. M. Gold, T. D. Charters and J. Ramsden. Automated contour mapping using triangular element data structures and an interpolant over each irregular triangular domain. "Proc. SIGGRAPH, 1977". *Comput. Graphics, A quarterly report of SIGGRAPH-ACM* **11 (2)** (1977), 170–175.

[GeSh90] M. K. Georges and M. S. Shephard. Automatic mesh generator for use in two-dimensional *h-p* analysis. *J. Comput. Civil Engrg.* **4** (1990), 199–220.

[Gilb79] P. D. Gilbert. New results in planar triangulations. M.S. Thesis, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1979.

[GKS92] L. J. Guibas, D. E. Knuth and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica* **7** (1992), 381–413.

[Greg75] J. A. Gregory. Error bounds for linear interpolation on triangles. *The Mathematics of Finite Element and Applications II,* J. R. Whiteman, ed., Academic Press, NY, 1975, 163–170.

[GuSt85] L. J. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graphics* **4** (1985), 74–123.

[Hans90] D. Hansford. The neutral case for the min-max triangulation. *Comput. Aided Geom. Design* **7** (1990), 431–438.

[HoLe88] K. Ho-Le. Finite element mesh generation methods: a review and classification. *Comput. Aided Design* **20** (1988), 27–38.

[Jans92] K. Jansen. One strike against the min-max degree triangulation problem. Manuscript, Fachbereich IV, Mathematik und Informatik, Universität Trier, Postfach 3825, W-5500 Trier, Germany, 1992.

[JKY92] J. W. Jaromczyk, M. Kowaluk and F. F. Yao. An optimal algorithm for constructing β-skeletons in L_p metric. To appear in *SIAM J. Comput.*

[Joe86] B. Joe. Delaunay triangular meshes in convex polygon. *SIAM J. Sci. Statist. Comput.* **7** (1986), 514–539.

[JoSi86] B. Joe and R. B. Simpson. Triangular meshes for regions of complicated shape. *Internat. J. Numer. Methods Engrg.* **23** (1986), 751–778.

[JoSi87] B. Joe and R. B. Simpson. Corrections to Lee's visibility polygon algorithm. *BIT* **27** (1987), 458–473.

[Kata88] J. Katajainen. The region approach for computing relative neighbourhood graphs in the L_p metric. *Computing* **40** (1988), 147–161.

[Klin80] G. T. Klincsek. Minimal triangulations of polygonal domains. *Ann. Discrete Math.* **9** (1980), 121–123.

[Lank69] P. M. Lankford. Regionalization: theory and alternative algorithms. *Geog. Anal.* **1** (1969), 196–212.

[Laws72] C. L. Lawson. Generation of a triangular grid with applications to contour plotting. Jet Propul. Lab. Techn. Memo. 299, Pasadena, CA, 1972.

[Laws77] C. L. Lawson. Software for C^1 surface interpolation. *Math. Software III*, J. R. Rice, ed., Academic Press, NY, 1977, 161–194.

[Lee83] D. T. Lee. Visibility of a simple polygon. *Comput. Vision, Graphics and Image Process.* **22** (1983), 207–221.

[Lee85] D. T. Lee. Relative neighborhood graphs in the L_1-metric. *Pattern Recognition* **18** (1985), 327–332.

[LeLi86] D. T. Lee and A. K. Lin. Generalized Delaunay triangulation for planar graphs. *Discrete Comp. Geom.* **1** (1986), 201–217.

[LeLi90] C. Levcopoulos and A. Lingas. Fast algorithms for greedy triangulation. *Lecture Notes in Comput. Sci.* **447** (1990), 238–250.

[Lind83] D. A. Lindholm. Automatic triangular mesh generation on surfaces of polyhedra. *IEEE Trans. Magnetics* **MAG-19** (1983), 2539–2542.

[Ling87] A. Lingas. A new heuristic for minimum weight triangulation. *SIAM J. Alg. Discrete Meth.* **8** (1987), 646–658.

[Lloy77] E. L. Lloyd. On triangulations of a set of points in the plane. *In* "Proc. 18th Ann. IEEE Sympos. Found. Comput. Sci., 1977", 228–240.

[MaSo80] D. W. Matula and R. R. Sokal. Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geog. Anal.* **12** (1980), 205–222.

[Meis75] G. H. Meisters. Polygons have ears. *Amer. Math. Monthly* **82** (1975), 648–651.

[MeSo92] E. A. Melissaratos and D. L. Souvaine. Coping with inconsistencies: a new approach to produce quality triangulations of polygonal domains with holes. *In* "Proc. 8th Ann. Sympos. Comput. Geom., 1992", 202–211.

[MiVa92] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. *In* "Proc. 8th Ann. Sympos. Comput. Geom., 1992", 212–221.

[Nadl85] E. J. Nadler. Piecewise linear approximation on triangulations of a planar region. Ph.D. Thesis, Div. Applied Math, Brown Univ., RI, 1985.

[NaSr91] L. R. Nackman and V. Srinivasan. Point placement for Delaunay triangulation of polygonal domains. *In* "Proc. 3rd Canadian Conf. Comput. Geom., 1991", 37–40.

[Olou91] A. A. Oloufa. Triangulation applications in volume calculation. *J. Comput. Civil Engrg.* **5** (1991), 103–119.

[OTZ88] S. Olariu, S. Toida and M. Zubair. On a conjecture by Plaisted and Hong. *J. Algorithms* **9** (1988), 597–598.

[PeKe87] G. Petrie and T. J. M. Kennie. Terrain modeling in surveying and civil engineering. *Comput. Aided Design* **19** (1987), 171–187.

[PlHo87] D. A. Plaisted and J. Hong. A heuristic triangulation algorithm. *J. Algorithms* **8** (1987), 405–437.

[PrSh85] F. P. Preparata and M. I. Shamos. *Computational Geometry—an Introduction.* Springer-Verlag, NY, 1985.

[QuSc90] E. Quak and L. L. Schumaker. Cubic spline fitting using data dependent triangulations. *Comput. Aided Geom. Design* **7** (1990), 293–301.

[Raja91] V. T. Rajan. Optimality of the Delaunay triangulation in $\mathbb{R}^d$. *In* "Proc. 7th Ann. Sympos. Comput. Geom., 1991", 357–363.

[Ripp90] S. Rippa. Minimal roughness property of the Delaunay triangulation. *Comput. Aided Geom. Design* **7** (1990), 489–497.

[Ripp92] S. Rippa. Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data. *SIAM J. Sci. Statist. Comput.* **13** (1992), 1123-1141.

[RiSc90] S. Rippa and B. Schiff. Minimum energy triangulations for elliptic problems. *Comput. Methods Appl. Mech. Engrg.* **84** (1990), 257–274.

[RuSe92] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional non-convex polyhedra. *Discrete Comput. Geom.* **7** (1992), 227–253.

[Saal91] A. Saalfeld. Delaunay edge refinements. *In* "Proc. 3rd Canadian Conf. Comput. Geom., 1991", 33–36.

[SaPe91] N. Sapidis and R. Perucchio. Delaunay triangulation of arbitrarily shaped planar domains. *Comput. Aided Geom. Design* **8** (1991), 421–437.

[Schö28] E. Schönhardt. Über die Zerlegung von Dreieckspolyedern in Tetraeder. *Math. Ann.* **98** (1928), 309–312.

[Schu87] L. L. Schumaker. Triangulation methods. *Topics in Multivariate Approximation*, C. K. Chui, L. L. Schumaker and F. I. Utreras, eds., Academic Press, 1987, 219–232.

[Seid88] R. Seidel. Constrained Delaunay triangulations and Voronoi diagrams with obstacles. *In* "1978–1988, 10-Years IIG", Inst. Inform. Process., Techn. Univ. Graz, Austria, 1988, 178–191.

[Shep88] M. S. Shephard. Approaches to the automatic generation and control of finite element meshes. *Appl. Mech. Rev.* **41** (1988), 169–185.

[ShHo75] M. I. Shamos and D. Hoey. Closest point problems. *In* "Proc. 16th Ann. IEEE Sympos. Found. Comput. Sci., 1975", 151–162.

[Sibs78] R. Sibson. Locally equiangular triangulations. *Comput. J.* **21** (1978), 243–245.

[StFi73] G. Strang and G. Fix. *An Analysis of the Finite Element Method.* Prentice-Hall, Englewood Cliffs, NJ, 1973.

[SuCh91] T. H. Su and R. C. Chang. Computing the constrained relative neighborhood graphs and constrained Gabriel graphs in Euclidean plane. *Pattern Recognition* **24** (1991), 221–230.

[Supo83] K. J. Supowit. The relative neighborhood graph, with an application to minimum spanning trees. *J. ACM* **30** (1983), 428–448.

[Tous80] G. T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition* **12** (1980), 261–268.

[Trac77] F. T. Tracy. Graphical pre- and post-processor for two dimensional finite element method programs. "Proc. SIGGRAPH, 1977". *Comput. Graphics, A quarterly report of SIGGRAPH-ACM* **11 (2)** (1977), 8–12.

[Voro07] G. Voronoi. Nouvelles applications des paramètres continus à la thèorie des formes quadratiques. Premier Mémoire: Sur quelques propriétés des formes quadratiques positives parfaites. *J. Reine Angew. Math.* **133** (1907), 97–178.

[Voro08] G. Voronoi. Nouvelles applications des paramètres continus à la thèorie des formes quadratiques. Deuxième Mémoire: Recherches sur les parallélloèdres primitifs. *J. Reine Angew. Math.* **134** (1908), 198–287.

[WaPh84] D. F. Watson and G. M. Philip. Survey on systematic triangulations. *Comput. Vision, Graphics and Image Process.* **26** (1984), 217–223.

[Waup92] R. Waupotitsch. Implementation and performance analysis of the minmax angle algorithm. M.S. Thesis, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1992.

[WGS90] F. W. Wilson, R. K. Goodrich and W. Spratte. Lawson's triangulation is nearly optimal for controlling error bounds. *SIAM J. Numer. Anal.* **27** (1990), 190–197.

[WWW85] P. Widmayer, Y. F. Wu and C. K. Wong. Distance problems in computational geometry with fixed orientation. *In* "Proc. 1st Ann. ACM Sympos. Comput. Geom., 1985", 186–195.

[Yao82] A. C. Yao. On constructing minimum spanning trees in k-dimensional space and related problems. *SIAM J. Comput.* **11** (1982), 721–736.

Vita

Tan, Tiow Seng was born on 2 November 1962 in Muar, Johor, Malaysia. He was brought up in the neighboring country, Singapore. He received a B.Sc. majoring in Computer Science and Mathematics from the National University of Singapore in April 1984. He also received the Data Processing Managers' Association Award (1984). Upon graduation, he was employed by the National Computer Board, Singapore, as an Information System Officer at the Ministry of Health.

In November 1985, Tan returned to the National University of Singapore as a Senior Tutor in the Department of Information Systems and Computer Science. He then finished a M.Sc. before going on a study leave at the University of Illinois in January 1988. His study leave was supported by a National University of Singapore Oversea Graduate Scholarship.

Tiow-Seng completed a Ph.D. in Computer Science in December 1992. He also received the C. W. Gear Outstanding Graduate Student Award (1992). He has since returned to the Department of Information Systems and Computer Science, National University of Singapore, Lower Kent Ridge Road, Singapore 0511, Republic of Singapore (email: tants@iscs.nus.sg).