May 2017

Degree Constrained Triangulation of Annular Regions and Point Sites

Bhaikaji Gurung
University of Nevada, Las Vegas, bhaikaji505@gmail.com

DEGREE CONSTRAINED TRIANGULATION OF ANNULAR REGIONS AND POINT SITES

by

Bhaikaji Gurung

Bachelor of Engineering (B.E.)
Pulchowk Campus, Lalitpur
2010

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2017

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 21, 2017

This thesis prepared by

Bhaikaji Gurung

entitled

Degree Constrained Triangulation of Annular Regions and Point Sites

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Laxmi Gewali, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

John Minor, Ph.D.
Examination Committee Member

Justin Zhan, Ph.D.
Examination Committee Member

Henry Selvaraj, Ph.D.
Graduate College Faculty Representative

Abstract

Generating constrained triangulations of point sites distributed in the plane is a significant problem in computational geometry. We present theoretical and experimental investigation results for generating triangulations for polygons and point sites that address node degree constraints. We characterize point sites that have almost all vertices of odd degree. We present experimental results on the node degree distribution of Delaunay triangulations of point sites generated randomly. Additionally, we present a heuristic algorithm for triangulating a given normal annular region with an increment of even degree nodes.

Acknowledgements

First and foremost I like to thank my thesis advisor Dr. Laxmi Gewali of the Department of Computer Science at the University of Nevada, Las Vegas for his continuous mentoring and guidance throughout the process of writing this thesis. Prof. Gewali was always there to solve my queries and to help me refer appropriate research papers and related resources. He led me in the right direction whenever he thought I needed it.

Apart from my advisor, I would like to thank the rest of my thesis committee: Prof. John Minor, Prof. Justin Zhan, and Prof. Henry Selvaraj, for their encouragement, and insightful comments.

Finally, I must express gratitude to my family for their patience, support, and encouragement during my pursuit of master degree. Appreciation is also extended to friends, seniors, and juniors for providing me with support and continuous encouragement throughout my years of study. Without all of them this accomplishment would not have been possible.

Bhaikaji Gurung

University of Nevada, Las Vegas
May 2017

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Triangulation of a given set of point sites in the plane is the process of connecting them by line segments so that the region populated by point sites is tiled by triangles. Each triangle of the triangulation is also called a face. The line segments connecting point sites are called edges. In a triangulated mesh, no two edges can intersect in their interior. The problem of composing a triangulation for a given group of nodes has been extensively investigated in computational geometry literature [dBvKOS97, O'R98].

Triangulation algorithms have been applied widely in several scientific and engineering disciplines including: (i) geographic information system (GIS), (ii) computer graphics, (iii) image processing, (iv) finite element method, and (v) robotics.

In Geographic Information System (GIS), the surface of the terrain is modeled by using triangles whose vertices are located close together in regions of rapid slope change and sparse in regions where slope change is minimal. In finite element methods (FEM), the computational domain is partitioned into a triangular mesh to obtain an approximate solution for heat flow and fluid flow problems. In computer graphics and image processing, the surface of a 3D object is modeled by a triangular mesh which is processed in streamline form to construct and transmit an image of the object model.

A set of points in the plane can be partitioned into triangles in many ways [O'R98]. In fact,it grants exponentially many ways [O'R98]. One of the most widely used triangulation methods is the Delaunay triangulation which satisfies certain proximity properties [O'R98]. For applications

in GIS, a special kind of triangulation called triangulated irregular network (TIN) is used. In TIN representation, the terrain surface is modeled by irregularly placed point sites which are arranged in a mesh of non-overlapping triangles. Each point site in a TIN network has three-dimensional coordinates.

The notion of 'quality' has been considered in triangulation networks. A triangulation is said to be of better quality if it has a large portion of triangles with an *aspect ratio* close to 1. It is remarked that the aspect ration *asp(t)* of a triangle t is defined in terms of the smallest enclosing rectangle *er(t)*. Specifically, *asp(t)* is the ratio of the width (smaller side) to length (larger side) of *er(t)*. If the aspect ratio is 1 then the smallest enclosing rectangle is a square. When a triangular mesh has a large portion of triangles with a high aspect ratio then such mesh can lead to increased accuracy in the approximate solution of a partial differential equation in finite element methods [Hut04].

Triangulation networks are used in computer graphics and image processing for rendering (generating an image from a geometric model) and transmission (transferring an object model). In such applications, it is necessary to construct a sequence of adjacent (edge sharing) triangles to form a "strip" [AHMS96]. Such a strip can be transmitted and rendered very efficiently. A triangulation mesh whose triangles can be arranged in one strip is called a Hamiltonian triangulation. Generating Hamiltonian triangulation of a given group of nodes is a challenging problem, and some noticeable results are reported in [AHMS96].

Another interesting problem in triangulation is the generation of minimum weight triangulation which is a triangulation that minimizes the total length of triangle edges. Minimum weight triangulation was the enduring problem in the area of computational geometry. In 2008, Mulzer and Rote [MR08] established that this problem is NP - Hard.

An interesting variation of a triangulation problem is the degree-constrained triangulation which seeks to construct a triangulated polygon with a large portion of even-degree nodes [PRVU10].

An overview of this thesis is as described in the following lines. In Chapter 2, we mention a cursory review of characterization and algorithm development in relation to the problem of generating a triangulated polygon with a large portion of an even degree nodes. In Chapter 3, we introduce

one of the main contributions of this thesis. In this chapter, we describe the characterization of an even degree triangulation in an annular region. We point out an $O(n^2)$ time algorithm for triangulating an annular region with a large portion of even degree vertices. In Chapter 4, we deliver an experimental investigation of degree distribution in Delaunay triangulations whose point sites are randomly generated. For this experimental investigation, we use a program in Java that allows the user to generate point sites by mouse clicks or to accept previously constructed point sites from a file. Results on even degree nodes for various values of node numbers n are presented. Lastly, in Chapter 5, we analyze: (i) discussions on the results obtained from experimental investigation, (ii) remarks on the algorithms developed in Chapter 3, and (iii) possible extensions and scope for future investigations on degree constrained triangulation.

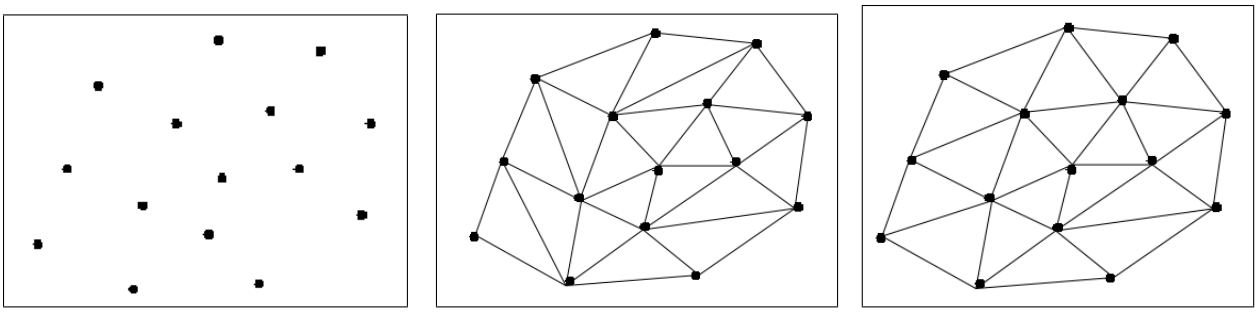
Chapter 2

Literature Review

In this chapter, we first present an overview of generating triangulation meshes for point sites distributed in the plane. We then review important results on generating triangulations satisfying node-degree constraints.

2.1 Triangulation

Given a set S of n point sites p_0, p_1,..., p_{n-1}, a triangulation formed by these sites is a partitioning of its convex hull such that: (i) the region inside the convex hull is partitioned into triangles and (ii) only the given point sites can be the vertices of the triangles. Two different triangulations of the same point sites is illustrated in Figure 2.1.

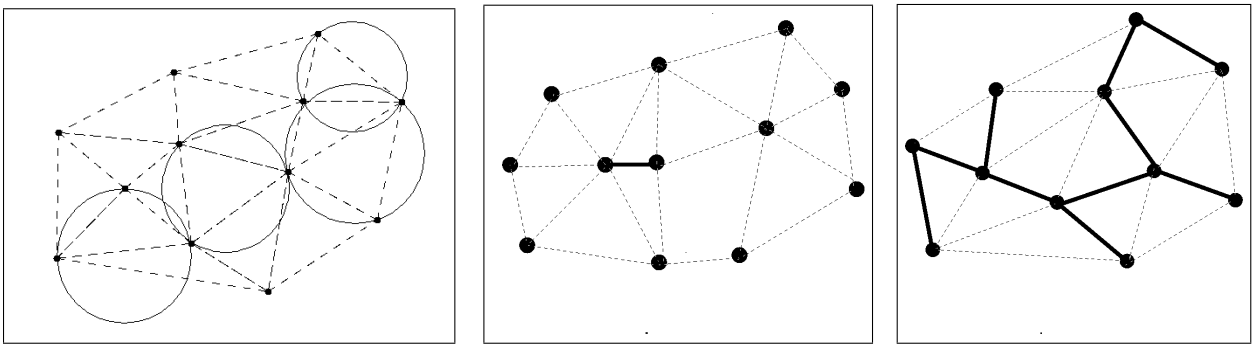


(a) (b) (c)

Figure 2.1: Two different triangulations

It is known that a given set S of n point sites admits exponentially many triangulations [dBvKOS97, O'R98]. While there are so many ways to triangulate a given group of nodes in 2D, a special type of triangulation called Delaunay triangulation [O'R98], has been considered extensively for many practical application [dBvKOS97].

Delaunay triangulation satisfies many interesting properties that include: (i) empty circle property, (ii) closest pair property, (iii) smallest angle property, and (iv) minimum spanning property. Specifically, the empty circle property states that the circle passing through any two/three point sites of a Delaunay triangle does not contain any other sites. This is exemplified in Figure 2.2a. The closest pair of sites in a set of points are neighbors in the Delaunay triangulation which is depicted in Figure 2.2b. The minimum spanning tree of a set of points is a subgraph of the Delaunay triangulation which is shown in Figure 2.2c.



(a) (b) (c)

Figure 2.2: Illustrating various properties of Delaunay Triangulation

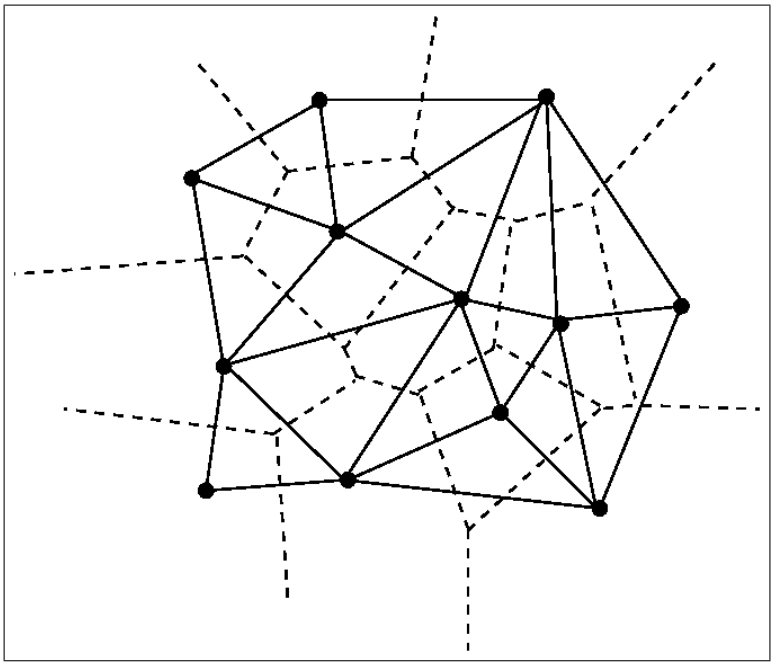


Figure 2.3: Delaunay Triangulation and Voronoi Diagram

Delaunay Triangulation is known to be the dual graph of Voronoi Diagram [O'R98]. It is noted that Voronoi Diagram of a given set of n point sites divides the plane into n regions called Voronoi cells. In each Voronoi cell *V(i)* for point site p_i, any point in V(i) is closer to p_i than the other point sites [O'R98]. Figure 2.3 depicts the overlay of Voronoi diagram and Delaunay triangulation. The Voronoi diagram can be computed in $O(nlogn)$ time by using a plane sweep algorithm developed by Steven Fortune [For87]. Due to the dual relationship between Voronoi diagram and Delaunay triangulation, an algorithm for constructing the Voronoi diagram can be used to obtain Delaunay triangulation within the same time complexity i.e. $O(nlogn)$.

2.2 Even Degree Triangulation

A triangulation of n nodes is called an **even triangulation** if all vertices are of even degree. Figure 2.4 shows an example of even triangulation.

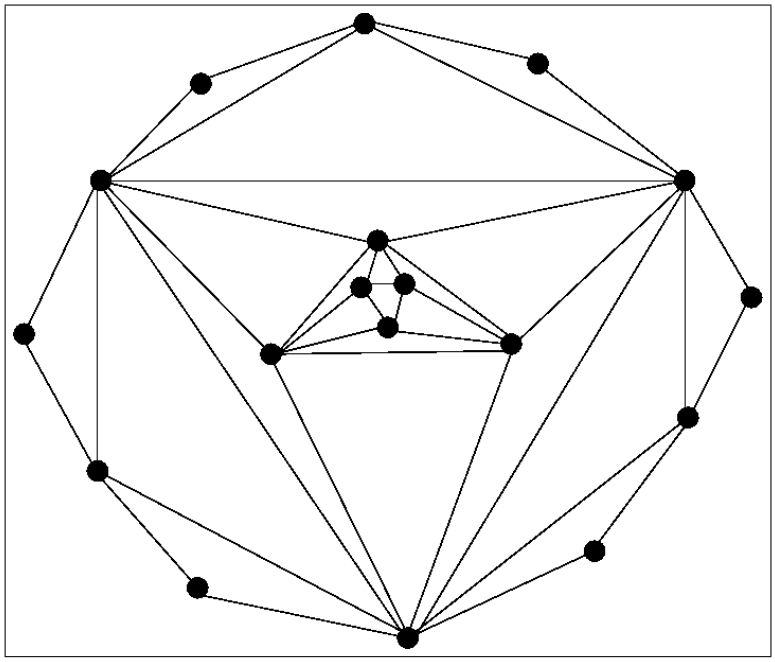


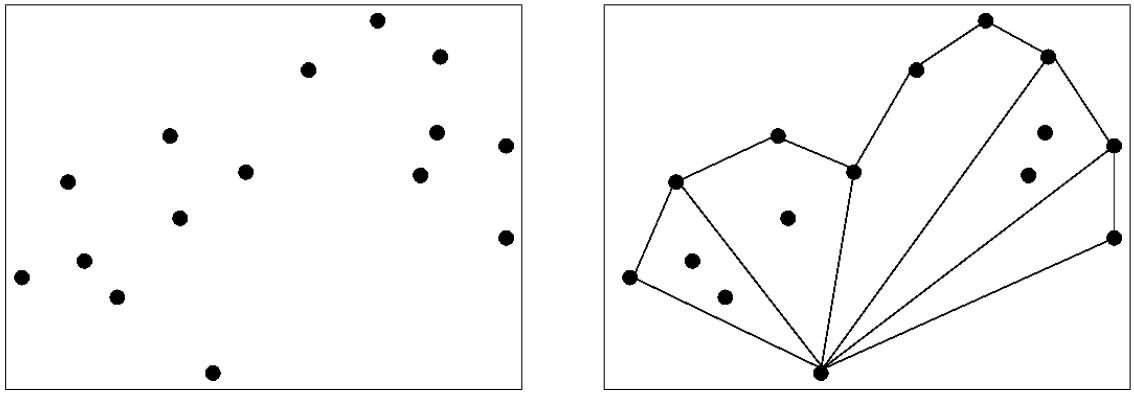
Figure 2.4: Illustrating an even-degree triangulation

Remark 2.1 *When considering even triangulation most authors [O'R98] consider the triangulation inside the convex hull of the given point sites.*

Some of the first results on even degree triangulation were reported in [AHH$^+$09, HK96, PRVU10]. It has been established that any bipartite 2-connected planar graph can be modified to a triangulation which is 3-colorable. This result can be applied for guard placement in orthogonal polygons

[HK96].

A notable result on even triangulation with a large percentage of even degree vertices was reported by Pelaez et. al. [PRVU10]. In this paper, it is shown that any given group of nodes can be partitioned into triangles to have at least $\lfloor \frac{2n}{3} \rfloor$-3 nodes with even degree. The method essentially partitions n input points sets into disjoint sectors S_1,S_2,...S_k originating at the point with the lowest y-coordinate. The sector-partitioning is such that each sector except the last one (S_k) contains exactly four points (not including sector source point). An instance of sector partitioning is shown in Figure 2.5.



(a) (b)

Figure 2.5: Illustrating Sector Partitioning

The method presented in [PRVU10] processes points in each sector in the order S_1, S_2,...,S_k. During each sector processing, edges are added to obtain a triangulation that tends to raise the count of even degree vertices. When processing points in sector S_i, the connectivity for the points in all sectors S_1,S_2,...,S_i are examined. By arguing complicated case analysis, it is shown that the resulting triangulation contains at least $\lfloor \frac{2n}{3} \rfloor$-3 vertices with an even degree.

2.3 Even Triangulation of polygons

Not all polygons can be evenly triangulated [Gya12]. In fact, there are polygons whose triangulation does not have any node of even degree. A long spiral polygon can be constructed in such a way that it admits only one triangulation and not all vertices are of even-degree. This is shown in Figure 2.6.

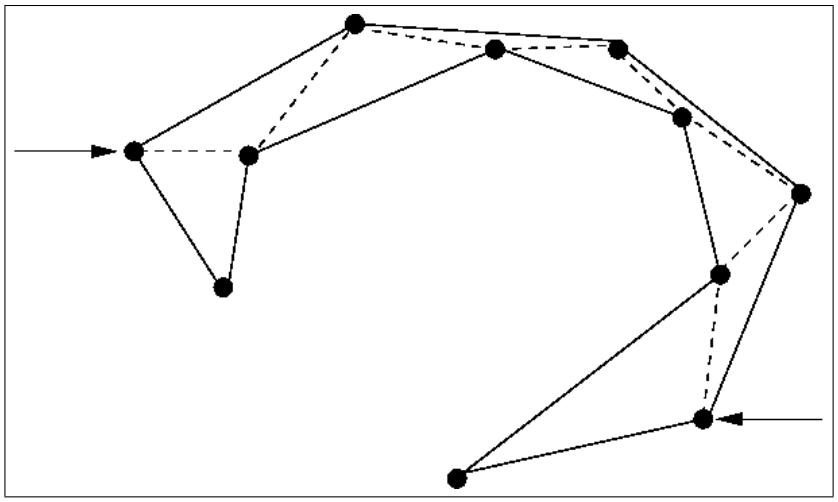


Figure 2.6: No Even Degree Triangulation

This example shows that a polygon of n vertices can be constructed to have exactly two vertices of odd degree. Convex polygons can be partitioned into triangles to have almost all even degree nodes. This result is reported in [Gya12]. It is shown that any convex polygon of n vertices where n is a multiple of 3 admits a triangulation with all even degree vertices. This is established in [Gya12] by first quadrangulating the convex polygon and then partitioning each quadrilateral into triangles forcing even degree vertices.

Chapter 3

Generating Even Triangulations

In this chapter new approaches for developing efficient algorithms for generating triangulations that tend to have a large percentage of even-degree nodes are presented.

3.1 Preliminaries

Consider a set S of point sites p_1, p_2,..., p_n in the 2D plane, some of which are connected to form a planar straight line graph (**pslg**) as shown in Figure 3.1.

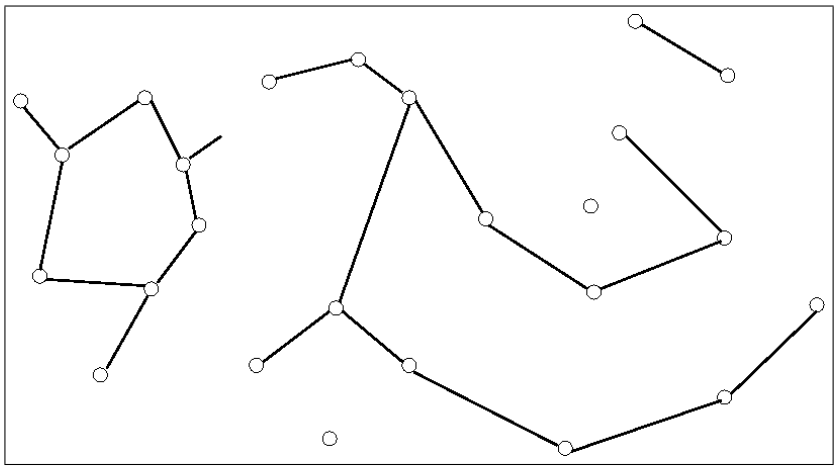


Figure 3.1: A planar straight line graph

A pslg can be converted into a triangulation by progressively adding triangles and edges. In doing so we need to add more to the even degree vertices. In some cases, the added triangle can raise the count of even-degree vertices while in other cases the number could decrease. For instance, consider the situation in Figure 3.2 which is obtained by adding four edges (shown dotted) to the

graph in Figure 3.1. The 2 dotted edges added to the top-left part increase the total count of even degree vertices by 2. On the other hand, the two dotted edges added in the bottom part actually decrease the total count of even degree vertices by 2. This gives us a hint on characterizing new edges that increase or decrease the count of even degree vertices.

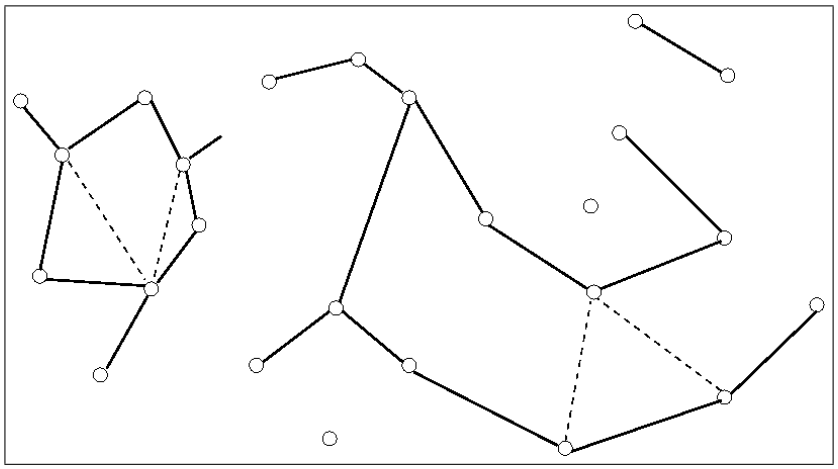


Figure 3.2: Adding Valid Chains to PSLG

Definition 3.1 *The **even potential** of a triangulation is the total count of even degree vertices. A triangulation of a high even potential will have a large portion of even degree vertices.*

Definition 3.2 *(**Valid chain**) A chain p_{i_1},p_{i_2},...,p_{i_k} is called **valid** if none of the edges of the chain intersect with the existing edges of the pslg.*

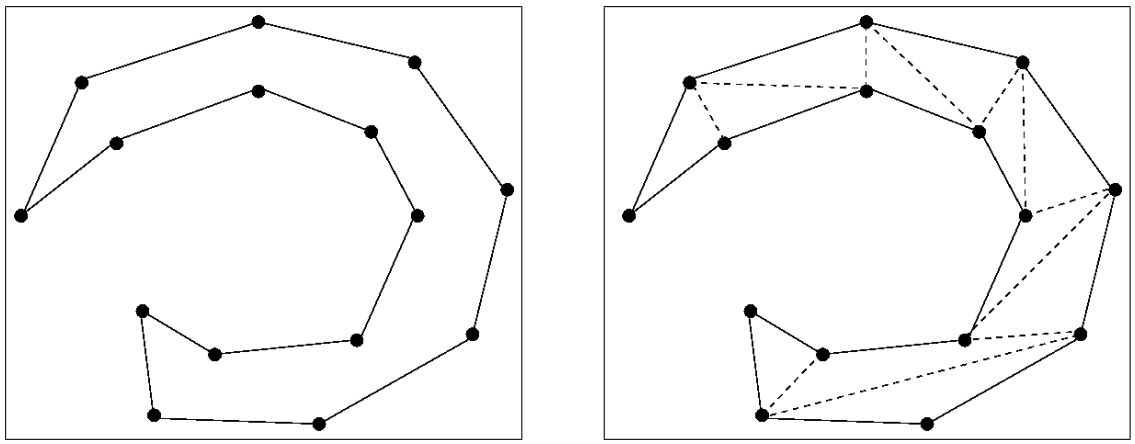
Now we examine the requirements that do not decrease the even potential when valid chains are added to a given pslg.

Property 3.1 *Let $ch_1 = p_{i_1}$,p_{i_2},p_{i_3},...,p_{i_k} be a sequence of non-consecutive vertices of the pslg. Adding ch_1 to the pslg increases the even potential of the new pslg if the degrees of both extreme vertices (before adding the chain) are odd. If only one of the p_{i_1} or p_{i_k} vertex is odd then the even potential stays the same.*

Property 3.2 *If the degrees of both vertices p_{i_1} and p_{i_k} are even then the even potential will decrease as ch_1 is added to the pslg.*

3.2 Triangulation of Spiral Polygons

A polygon whose boundary consists of exactly one **convex chain** and one **concave chain** is referred to as a **spiral polygon**. Note that a convex chain of a polygon has all its internal angles less than 180^0. Similarly, a concave chains internal angles are greater than 180^0. Figure 3.3a shows a spiral polygon in which the size of its convex chain and concave chain are comparable. It is remarked that when we use the term 'concave chain' or 'convex chain' of a spiral polygon it is understood to be maximal chains.



(a) (b)

Figure 3.3: Triangulating a Normal Spiral Polygon

A spiral polygon having convex and concave chains of comparable size can be triangulated in a "back and forth" manner to have the most vertices of even degree. In the back and forth approach for triangulation, the concave chain and convex chain are connected by picking vertices from each alternatively so that (except for start and end vertices) all vertices are of even degrees. Figure 3.3b shows the triangulation based on the back and forth approach.

3.3 Triangulation of Annular Regions

An *annular region* is formed by two closed circular chains. We can also think of an annular region as a polygon with one hole. We now consider the problem of triangulating an annular region to maximize the number of even degree vertices.

We first examine some extreme instances of an annular region. We consider the situation when the

interior chain is just a triangle and the outside chain consists of many vertices. In our investigation, we consider both interior and outside chains to be convex. This is illustrated in Figure 3.4. The annular region can be partitioned into six convex regions by using two diagonals emanating from the vertices of the triangle, which are shown by dashed lines in the Figure 3.4. Out of the six regions, three are triangles and the other three are convex. The convex regions can be partitioned into triangles to have all even-degree nodes by using the algorithm reviewed in Chapter 2. In the resulting triangulation, except for six vertices in the outer chain, all vertices are of even degree. This is desribed in Lemma 3.1.

Lemma 3.1 *If the inner chain is just a triangle, then the annular region can be triangulated to have at most 6 vertices of odd degrees.*

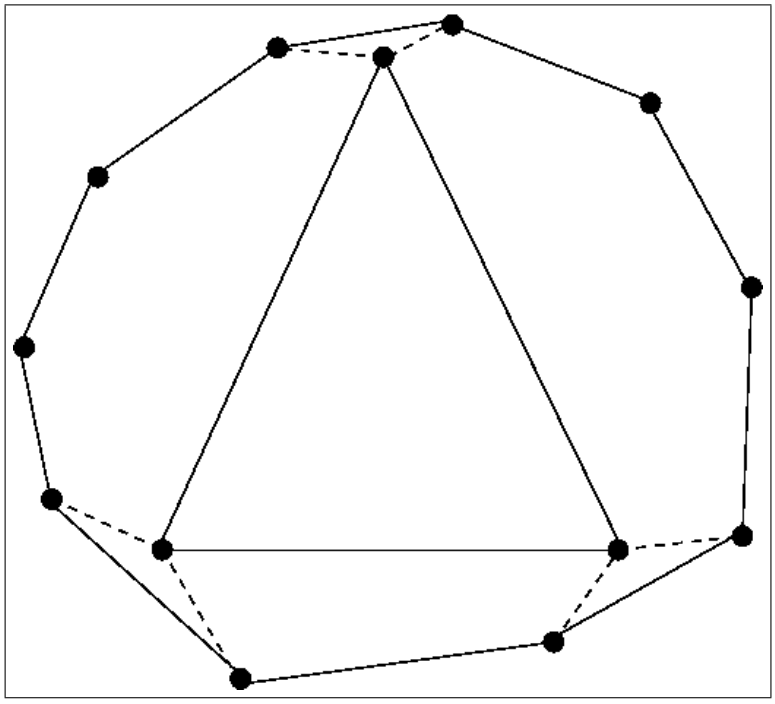


Figure 3.4: Illustrating Lemma 3.1

Remark 3.1 *If the outer chain is also a triangle, then the triangulation becomes such that all six triangles are of even degree.*

Next, we examine the situation when the outside chain is a triangle and the inside chain has many vertices. This is illustrated in Figure 3.5. Any triangulation of such an annular region is such that only three vertices in the inner chain are of even degree. The degree parity of the vertices of the

outer chain depends on the visibility relationship between inner and outer chain. This is described in Lemma 3.2.

Lemma 3.2 *For any given n>3, we can always construct an annular region that admits triangulation with no more than 6 vertices with the even degree.*

Proof: Construct a big enclosing triangle that encloses a convex polygon with *n-3* vertices. Observe that any triangulation of the annular region must have edges with one vertex in an outer triangle and one vertex in an inner convex polygon. Furthermore, at most two diagonals can be incident upon a vertex of the inner chain. Let w_i, w_j, and w_k be the vertices of the inner chain where two diagonals are incident as shown in Figure 3.5. On all other vertices of the inner chain, exactly one diagonal will incident - making all of them of odd degree. Hence at most three vertices of the outer chain and only three vertices of the inner chain can be of even degree which implies that no more than six vertices are of even degree. ■

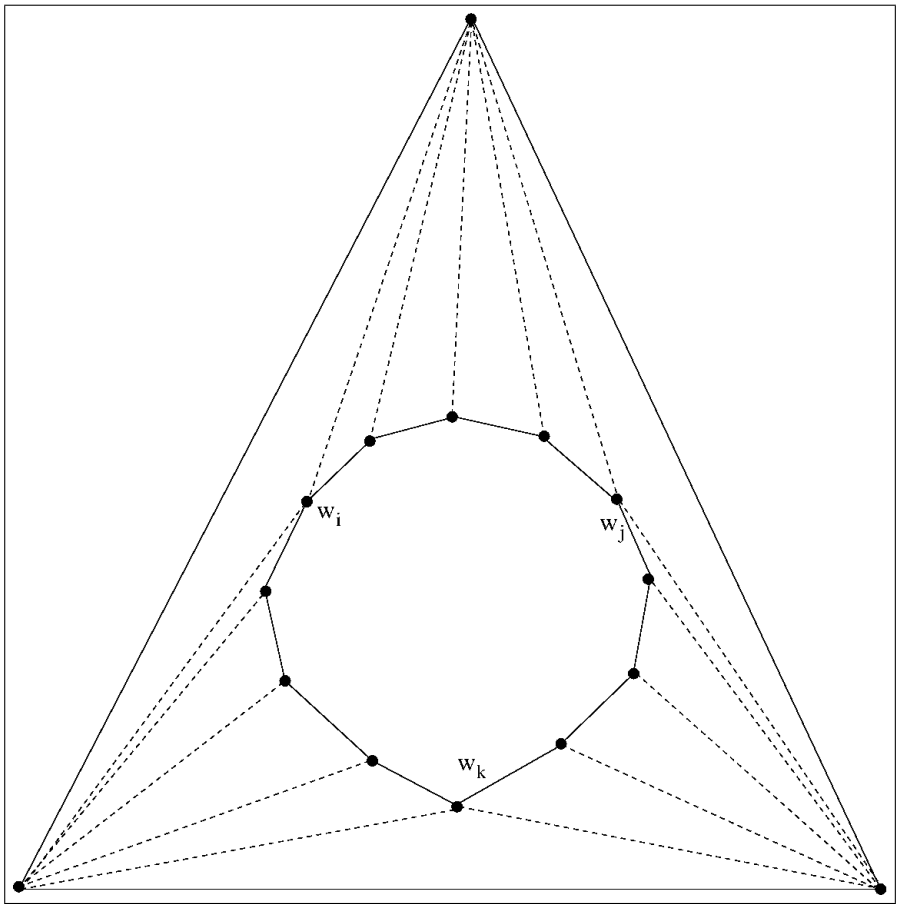


Figure 3.5: Illustrating Lemma 3.2

The annular region formed by two convex chains of a comparable number of vertices in each chain is referred to as a **normal annular region**. An instance of a normal annular region is illustrated in Figure 3.6, where the outer chain has 16 vertices and the inner chain has 14 vertices. An efficient heuristic algorithm for triangulating normal annular regions with an increment in the number of even degree nodes is described in the next section.

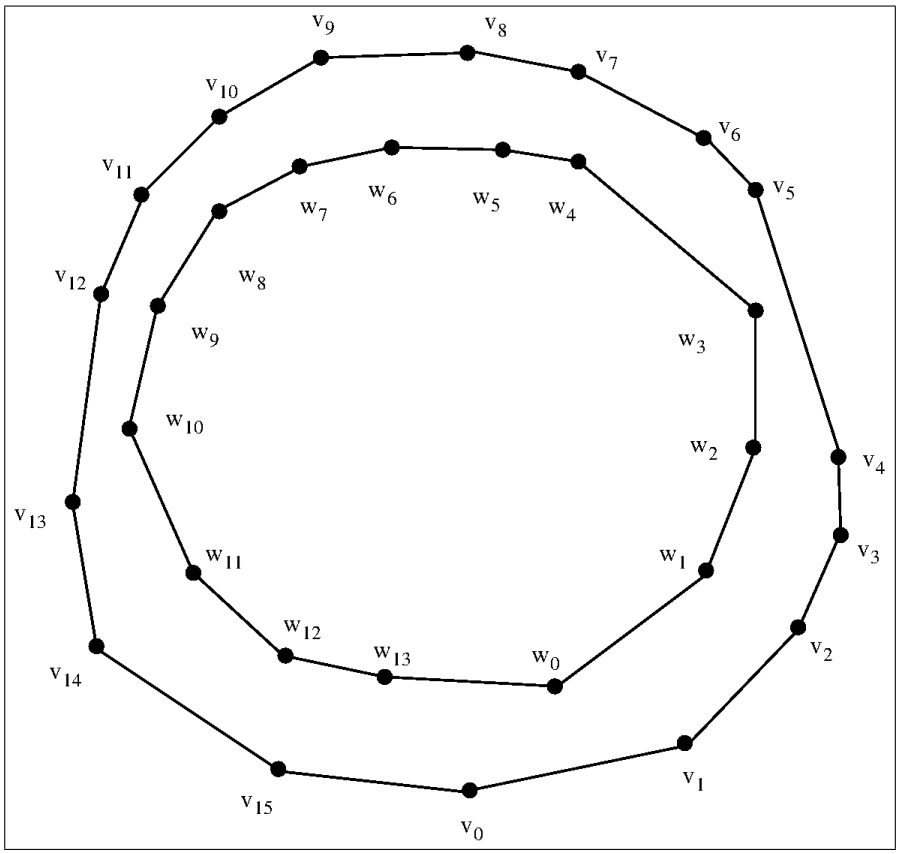


Figure 3.6: Normal Annular Region

3.4 Development of Alternate Marching Algorithm

Let v_0,v_1,v_2,..., v_{n_1} be the vertices in the outer boundaries (counterclockwise order). Similarly, let w_0, w_1 , w_2 ,..., w_{n_2} be the vertices in the inner boundary in counterclockwise order. In the **Alternate Marching algorithm (AM algorithm)** we propose, a chain is constructed connecting vertices of outer and inner boundary alternately as far as possible to obtain triangles whose vertices, except the first triangle and last triangle, will be of even degree. Such a chain is referred to as Alternating Max Chain **(AM-Chain (v_i))**, where v_i is the starting vertex of the chain. This is illustrated in Figure 3.7, where AM-Chain (v_0) is the chain <v_0,w_0,v_1,w_2,...,w_3,v_4>. The chain cannot proceed beyond w_3 as (v_4,w_4) intersects with the inner boundary. The next connecting chain is **AM-Chain(v_5)**.

In some unusual cases, vertex v_5 may not be visible to vertex w_4. This happens if v_5 is very close to v_4 making line segment v_5w_4 intersect with the inner boundary. If many vertices are clustered and close to v_4 then all vertices in the cluster may not be visible to w_4. In such cases, enough vertices starting from v_4 are skipped to find the first vertex in the outer boundary visible to w_4. For the clarity of presentation, we assume, without loss of generality, that skipping one vertex is enough. This process of constructing maximal alternating chain is continued until the starting vertex v_0 is not reached again. It is observed that for AM-Chain(v_0), the first vertex v_0 and last vertex v_4 are of odd degree and all other vertices w_0,v_1,w_1,v_2,w_2,v_3,w_3 are of even degree. In Figure 3.7a, the next alternating chain starts at v_5 which is <v_5,w_4,v_6,w_5,...,w_{10},v_{12}>. The next alternating chain is <v_{13},w_{11},v_{14},w_{12},...,w_{13},v_0>. Thus the annular chain is covered with 3 maximal alternating chains. The interior vertices of all alternating chains are of even degree. The region between two alternating chains can be a polygon with more than three vertices which we call a *squeezed region*. In the example (Figure 3.7a) there are two squeezed regions each with four vertices. The squeezed regions are triangulated by using any polygon triangulation algorithm [O'R98].

The algorithm can be formally sketched and is listed as Algorithm 1. Algorithm 1 invokes function ConstructAM-Chain (v_a, CH_I, CH_O) which triangulates the region between two chains CH_O and CH_I starting at v_a by constructing AM-Chain(v_0). It is noted that AM-Chain(v_0) is maximal.

Algorithm 1: Triangulate Annular Region **R**

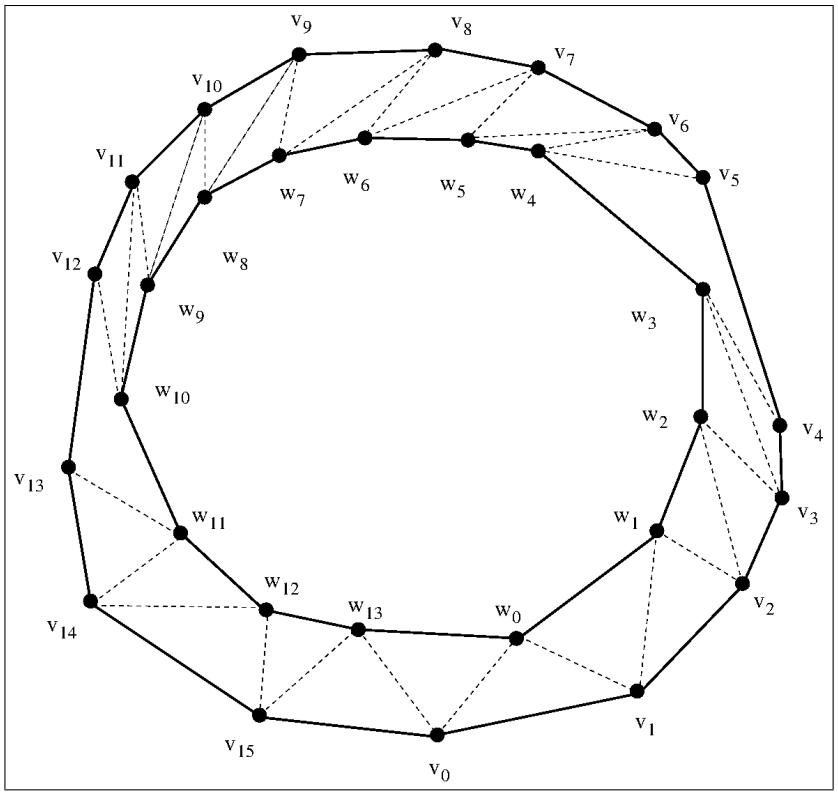
Input: (i) Outer vertices v_0,v_1,v_2,..., $v_{n_1} = CH_O$
(ii) Inner vertices w_0, w_1 , w_2 ,..., $w_{n_2} = CH_I$

Output: Triangulation of Annular Region R with an increment of even degree nodes

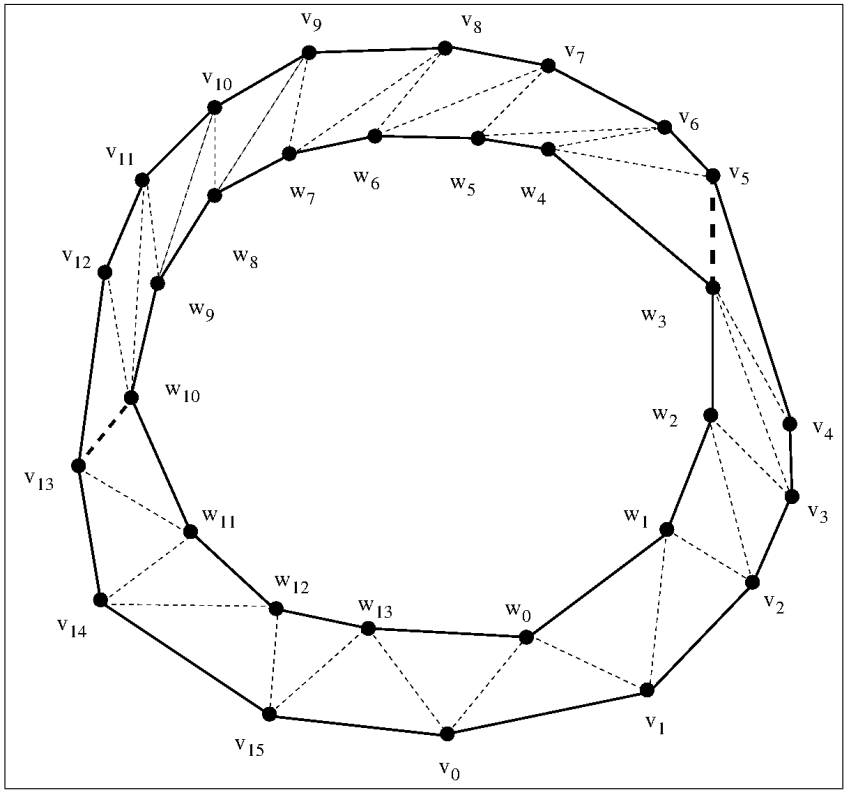
(i) Let v_{s_0} be the start vertex in the outer boundary.
(ii) Let w_{r_0} be the vertex closest to v_{s_0} in the inner boundary
(iii) Ch = ConstructAM-Chain(v_{s_0}, CH_I, CH_O)
(iv) Let the vertices of Ch be v_{s_0}, w_{r_0}, v_{s_0+1}, w_{r_0+1}, ..., $v_{s_0+t_0}$, $w_{r_0+t_0}$
(v) Output Ch
(vi) $i = 0$; $j = 0$;
while $v_{s_i+t_i} \neq v_{s_0}$ **do**
    Ch=ConstructAM-Chain($v_{s_i+t_i+1}$, CH_I, CH_O);
    Output Ch;
    $i = i + 1$; $j = j + 1$;
    Let the vertices of Ch be v_{s_i}, w_{r_j}, v_{s_i+1}, w_{r_j+1}, ..., $v_{s_i+t_i}$, $w_{r_j+t_i}$
    Triangulate squeezed regions by using standard polygon triangulation algorithms

```
1  Function ConstructAM-Chain (v_a, CH_I, CH_O)
2      Ch = φ;
3      Add v_a to Ch;
4      s = a;
5      Let w_b be the unprocessed vertex of CH_I closest to v_a
6      if w_b is null then return Ch;
7      Add w_b to Ch;
8      Done = false;
9      s = s + 1;
10     while not Done do
11         if w_b is visible to v_s then
12             Add v_s to Ch;
13         else
14             Done = true;
15             return Ch;
16         b = b + 1;
17         if v_s is visible to w_b then
18             Add w_b to Ch;
19         else
20             Done = true;
21             return Ch;
22
```

The time complexity of Algorithm 1 depends on the way function ConstructAM-Chain (v_a, CH_I, CH_O) is implemented. A straightforward way of determining visibility between w_b and v_s (line 11) is to check the intersection between line segment (w_b,v_s) and inner chain CH_I. This approach works correctly but is not efficient. A better way is to make use of the visibility graph induced by the vertices of the annular region.



(a)



(b)

Figure 3.7: Triangulation of Annular Shape

The visibility graph generated by a collection of convex obstruction is a scrutinized hurdle in computational geometry [O'R98]. This concept can be briefly explained as follows. The collection of obstacle vertices give the vertex set V of the visibility graph *VG(V,E)*. If two vertices v_i, $v_j \in V$ can be connected by a line segment without intersecting any other obstacle then (v_i, v_j) is a visibility edge. The collection of all visibility edges is the set *E*. Figure 3.8. is an illustration of a visibility graph generated by polygonal obstacles enclosed in a rectangular box.

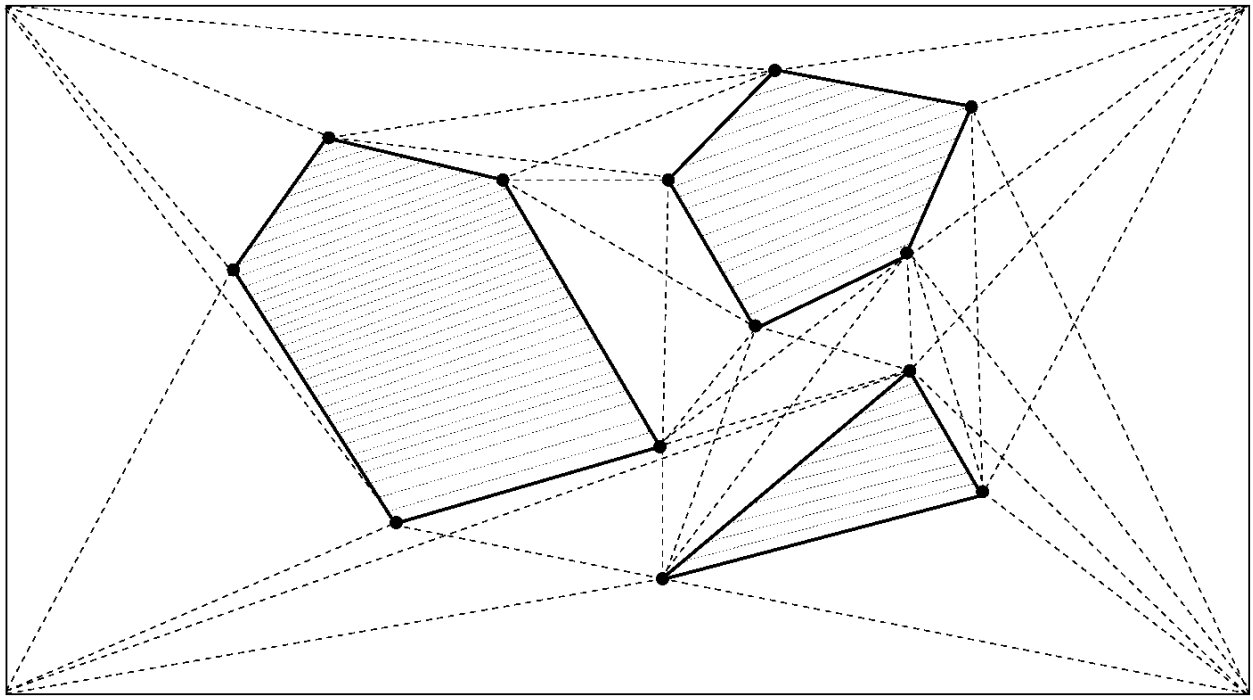


Figure 3.8: Visibility Graph of Polygonal Obstacles

A visibility graph for an annular region can be constructed similarly by connecting all visible vertex pairs which is shown in Figure 3.9. It is observed that triangulation diagonals are contained in the visibility graph of the annular region. From the visibility graph of the annular region a subset of edges can be carefully removed to obtain a triangulation that increases the number of even degree vertices. To efficiently implement line 11 of the function ConstructAM-Chain (v_a, CH_I, CH_O) we could use a precomputed visibility graph of the annular region.

The visibility graph of the annular region is depicted in Figure 3.9 where the visibility edges picked by Algorithm 1 are distinctly shown with thick edges.

Theorem 3.3 *Algorithm 1 can be implemented in $O(n^2)$ time, where n is the number of vertices in the annular region.*

Proof:

The visibility graph of a collection of polygonal obstacles can be computed in $O(|E|)$ time where $|E|$ is the number of visibility edges [GM91]. Step 1(i) and step 1(ii) can be done in time proportional to the degree of vertex v_{s_0} by checking the visibility edges emanating from v_{s_0} in the visibility graph. One call of function ConstructAM-Chain() takes time proportional to the size of the maximum alternating chain due to the availability of the precomputed visibility graph. Since no vertex is visited more than a constant time during the chain scan, the total time for executing Algorithm 1 is no more than the size of the visibility graph. Thus the total time is bounded by $O(|E|)$ which is $O(n^2)$ in the worst case.

■

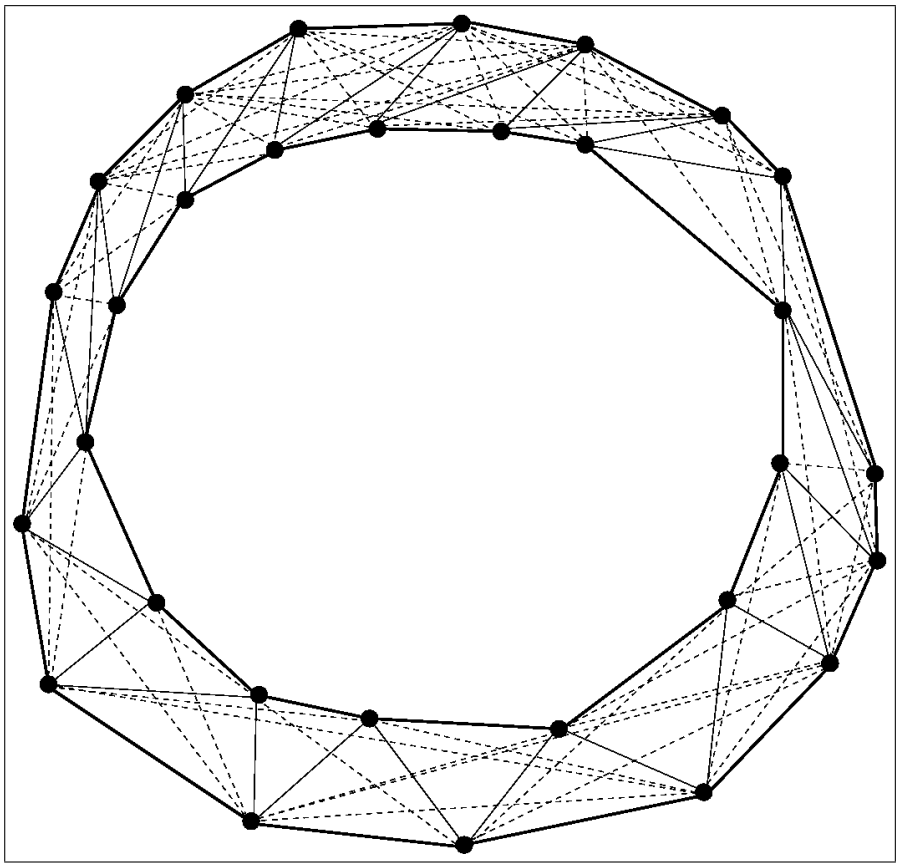


Figure 3.9: Visibility Graph of Annular Region and Triangulation Edges

Chapter 4

Implementation and Experimental Results

In this chapter, we describe a brief explanation of the implementation of algorithms for determining percentage distribution of even/odd node degrees for triangulation of point sites. Specifically, we consider a distribution of even/odd node degrees of the Delaunay Triangulation. The implementation is done in the Java programming language with a friendly graphical user interface (GUI), through which a user can enter both manually and randomly generated points. The GUI allows the user to edit location of points by mouse drag. The coordinates of the clicked or generated points are also displayed in a scrollable textbox.

Computational geometry library functions available from Joseph O'Rourkes' site[1] are used for performing various geometric computations that include: (i) Delaunay Triangulation, (ii) Convex Hull in 2D, (iii) intersection and related computation on line segments, rays, polygons, and (iv) Voronoi Diagram. We use the doubly-connected edge list representation [dBvKOS97] implemented by Gewali and his group at UNLV as a data structure for storing the Delaunay triangulation and its dual, the Voronoi diagram.

4.1 Interface Description

The front-end of the program is a GUI developed by using JFrame object imported from the java.swing package. The rectangular JFrame frame is structured into five regions which are: north,

[1] http://cs.smith.edu/ jorourke

east, west, south, and central. This partitioning is shown in Figure 4.1.

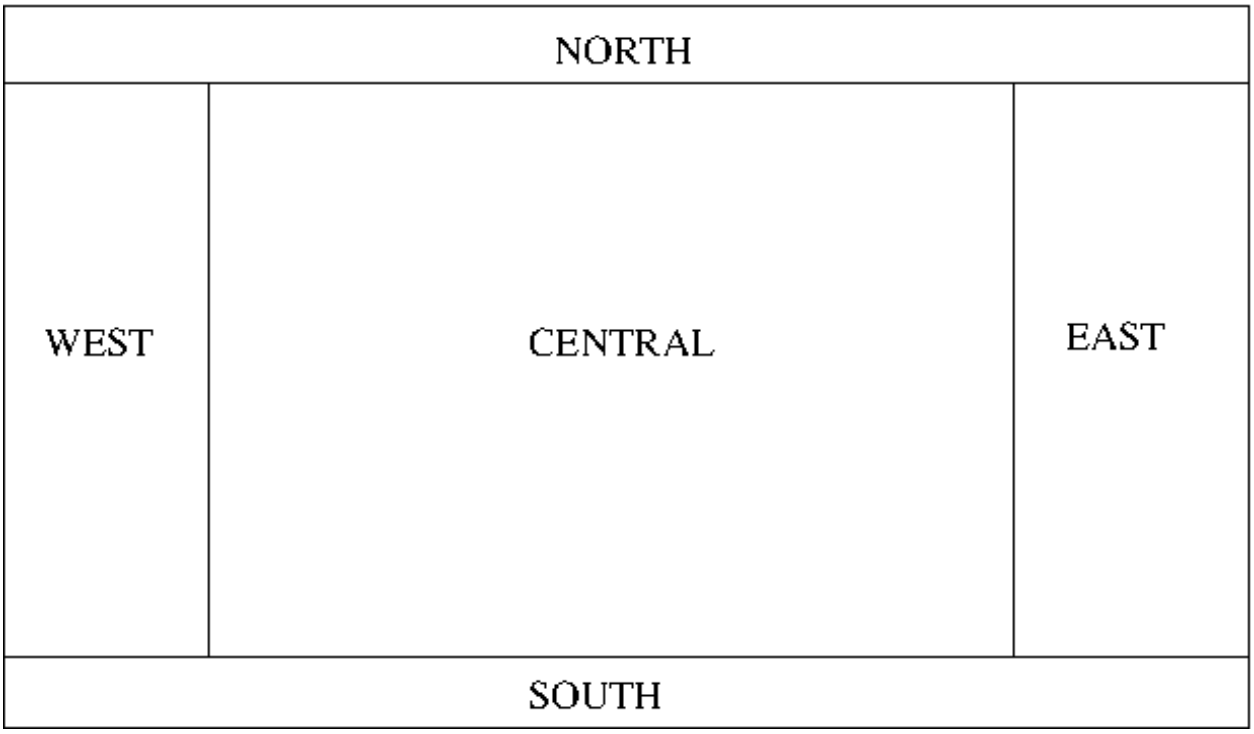


Figure 4.1: Graphical User Interface Layout

The rectangular region 'North' is used to hold a menubar of various file read/write operations. At present the menubar contains a dropdown list for reading and saving triangulations in doubly connected edge list (DCEL) form.

The 'West' region contains a Java panel which is used to hold various check boxes with text labels. Specifically, there are nine check boxes, seven of which can be checked to trigger geometric computation on points that include Delaunay triangulation, Voronoi diagram, Circumcircle test, spanning tree, and Hull wrap. Two check boxes are not used but can be adopted for future computation.

The 'East' region holds three sub-panels. The top sub-panel contains four Java button components with indicated functionalities. The middle sub-panel contains a scrollable text area which is used to display the co-ordinates of points entered by mouse clicks. The lower subpanel is left for

future use. A snapshot of the main GUI is shown in Figure 4.2.

Figure 4.2: Snapshot of Main GUI

4.2 Component Functionalities

The names of each interface component are chosen to indicate their intended functions. The names of the components are either given on themselves or as a text label next to them. The 'Draw Vertex' checkbox is such that when it is checked the mouse click will draw a point on the drawing canvas. Similarly, when the 'Triangulate' checkbox is checked the programs displays the Delaunay triangulation of the points displayed on the canvas.

The co-ordinates of the nodes drawn on the canvas are also shown in the scrollable text area on the east side of the GUI. These coordinates can be used by cut/paste operations on other programs. A short explanation of the functionalities of the GUI components is given in Table 4.1 - 4.4.

It is important to note that the read/save selection in the dropdown menu is used to save the entire data structure of the Delaunay triangulation and Voronoi Diagram of the points on the canvas. More specifically, a list of the vertices, faces, and half-edges of the Delaunay triangulation are saved on the file together with the co-ordinates of the point sites. This is done so that triangulation and Voronoi diagram need not be recomputed when previously stored points are read from a file.

The program allows computation of the Delaunay triangulation on-the-fly, which means if the location of a vertex is dragged by mouse then the triangulation is recomputed and displayed. This on-the-fly computation helps us examine the results of a network computation interactively. A few snapshots of the computation of even/odd degrees of triangulation are shown in Figure 3.6.

S.N.	**FileMenu Items**	**Actions**
1	Read Dcel File	needed to show dcel data from a file
2	Save Dcel File	needed to store dcel data for future use

Table 4.1: Description of FileMenu Item

S.N.	Check boxes	Actions
1	Draw Vertex	enable to draw point by mouse click
2	Edit Vertex	enable to relocate existing point by mouse drag
3	Triangulate	enable to trigger a display of Delaunay triangulation
4	Voronoi Network	enable to draw a dual graph of Delaunay triangulation
5	Minimum Spanning Tree	enable to draw a minimum spanning tree of a graph
6	Circumcircle Test	enable to test the empty circle
7	Hull-wrap	enable to draw a convex boundary of the point sites

Table 4.2: Description of Checkbox

S.N.	Buttons	Actions
1	Random Point Sites	puts set of point sites randomly
2	Refresh Canvas	puts co-ordinates on the canvas from textbox of vertex co-ordinates
3	Refresh Textbox	replaces node coordinates of textbox with that of current point sites
4	Clear Canvas	removes everything from the canvas

Table 4.3: Description of Button

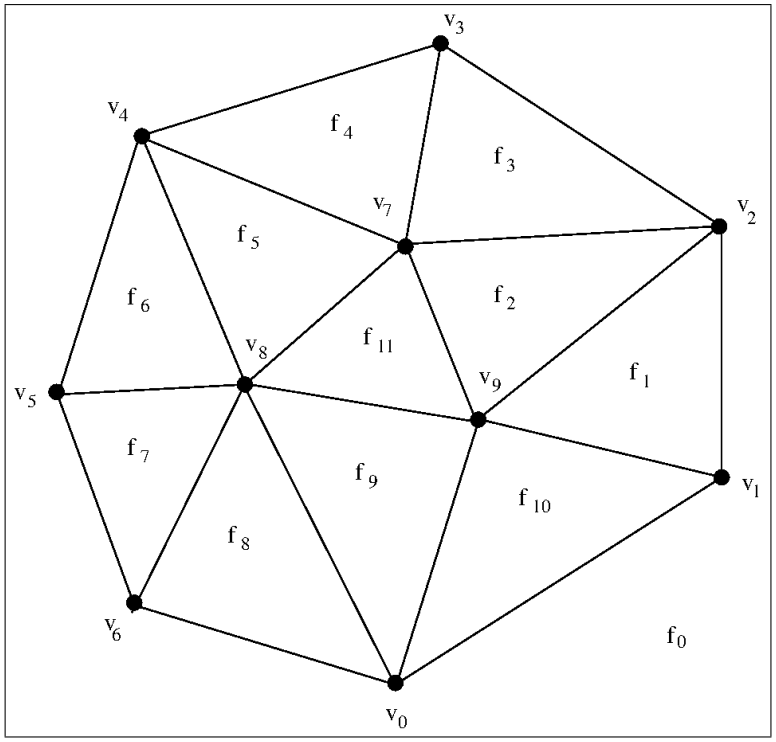
S.N.	Textboxes	Actions
1	Vertex Coordinates	shows the co-ordinates of point sites

Table 4.4: Description of Textbox

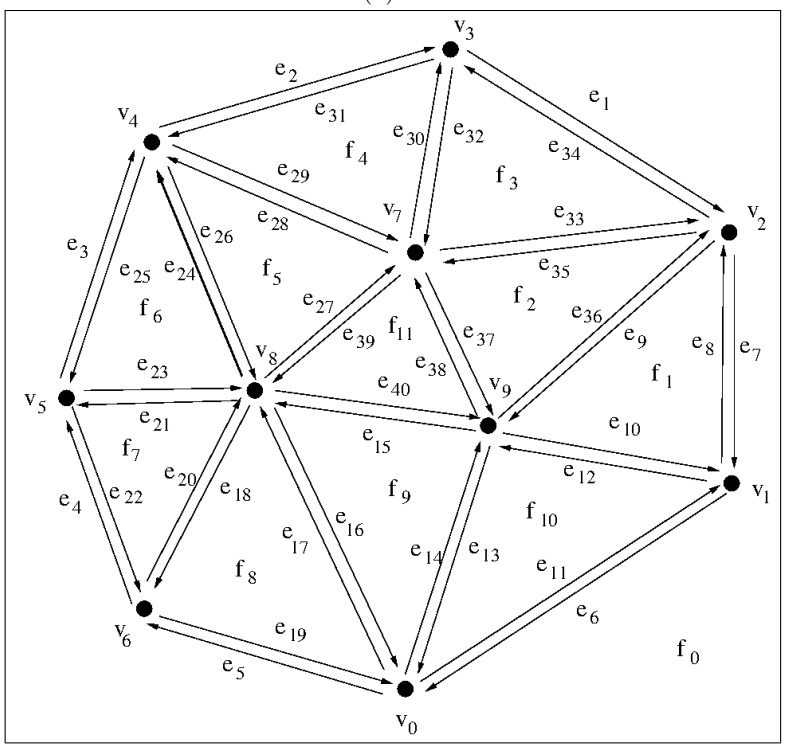
4.3 Results of Delaunay Triangulation

The Delaunay triangulation computed by the program is represented in a doubly connected edge list(dcel) data structure [dBvKOS97]. In a dcel data structure, a triangulation (in fact any planar graph) is viewed as consisting of three types of objects: (i) vertices, (ii) half-edges, and (iii) faces. We can describe this data structures with an illustrative example. Consider the Delaunay triangulation of 10 point sites as depicted in Figure 4.3a. To store it in a dcel data structure, each edge of the triangulation is viewed as a pair of directed half edges or twin half edges in opposite directions. The edge (v_2,v_3) of the Delaunay triangulation of Figure 4.3a. is shown as a pair of half edges e_1 and e_{34} (directed in opposite directions) in Figure 4.3b.

When all edges of Figure 4.3a are represented by a pair of twin half edges we get the structure shown in Figure 4.3b. By dcel convention, each triangle is bounded by three half-edges and can be traversed in a counterclockwise direction. It is noted that in addition to 10 triangular faces there is one unbounded face of size 7 which can be traversed in a clockwise direction.



(a)



(b)

Figure 4.3: DCEL

As explained in [dBvKOS97], a half edge has 5 properties: (i) previous half edge, (ii) next half edge, (iii) twin half edge, (iv) incident vertex, and (v) incident face. For instance, the 5 properties for edge e_9 are e_8(previous half edge), e_{10}(next half edge), e_{36}(twin half edge), v_2(incident vertex), and f_1(incident face).

In this way, a record of all half edges can be stored in a table as shown in Table 4.5.

Half Edge	**Previous**	**Next**	**Twin**	**Incident Vertex**	**Incident face**
e_1	e_2	e_7	e_{34}	v_3	f_0
e_2	e_3	e_1	e_{31}	v_4	f_0
e_3	e_4	e_2	e_{25}	v_5	f_0
e_4	e_5	e_3	e_{22}	v_6	f_0
e_5	e_6	e_4	e_{19}	v_0	f_0
e_6	e_7	e_5	e_{11}	v_1	f_0
e_7	e_1	e_6	e_8	v_2	f_0
e_8	e_{11}	e_{34}	e_7	v_1	f_1
e_9	e_8	e_{10}	e_{36}	v_2	f_1
e_{10}	e_9	e_8	e_{12}	v_9	f_1
e_{11}	e_{13}	e_{12}	e_6	v_0	f_{10}
e_{12}	e_{11}	e_{13}	e_{10}	v_1	f_{10}
e_{13}	e_{12}	e_{11}	e_{14}	v_9	f_{10}
e_{14}	e_{16}	e_{15}	e_{13}	v_0	f_9
e_{15}	e_{14}	e_{16}	e_{40}	v_9	f_9
e_{16}	e_{15}	e_{14}	e_{17}	v_8	f_9
e_{17}	e_{19}	e_{18}	e_{16}	v_0	f_8
e_{18}	e_{17}	e_{19}	e_{20}	v_8	f_8
e_{19}	e_{18}	e_{17}	e_5	v_6	f_8
e_{20}	e_{22}	e_{21}	e_{18}	v_6	f_7
e_{21}	e_{20}	e_{22}	e_{23}	v_8	f_7
e_{22}	e_{21}	e_{20}	e_4	v_5	f_7
e_{23}	e_{25}	e_{24}	e_{21}	v_5	f_7

Continued on next page

Table 4.5 – continued from previous page

Half Edge	**Previous**	**Next**	**Twin**	**Incident Vertex**	**Incident face**
e_{24}	e_{23}	e_{25}	e_{26}	v_8	f_6
e_{25}	e_{24}	e_{23}	e_3	v_4	f_6
e_{26}	e_{28}	e_{27}	e_{24}	v_4	f_5
e_{27}	e_{26}	e_{28}	e_{39}	v_8	f_5
e_{28}	e_{27}	e_{26}	e_{29}	v_7	f_5
e_{29}	e_{31}	e_{30}	e_{28}	v_4	f_4
e_{30}	e_{29}	e_{31}	e_{32}	v_7	f_4
e_{31}	e_{30}	e_{29}	e_2	v_3	f_4
e_{32}	e_{34}	e_{33}	e_{30}	v_3	f_3
e_{33}	e_{32}	e_{34}	e_{35}	v_7	f_3
e_{34}	e_{33}	e_{32}	e_1	v_2	f_3
e_{35}	e_{36}	e_{37}	e_{33}	v_2	f_2
e_{36}	e_{37}	e_{35}	e_9	v_9	f_2
e_{37}	e_{35}	e_{36}	e_{38}	v_7	f_2
e_{38}	e_{40}	e_{39}	e_{37}	v_9	f_{11}
e_{39}	e_{38}	e_{40}	e_{27}	v_7	f_{11}
e_{40}	e_{39}	e_{38}	e_{15}	v_8	f_{11}

Table 4.5: Record of Half Edges

Vertex	**x-coordinate**	**y-coordinate**	**Incident Half Edge**
v_0	x_0	y_0	e_{11}
v_1	x_1	y_1	e_8
v_2	x_2	y_2	e_{34}
v_3	x_3	y_3	e_{31}
v_4	x_4	y_4	e_{25}
v_5	x_5	y_5	e_{22}
v_6	x_6	y_6	e_{19}
v_7	x_7	y_7	e_{37}
v_8	x_8	y_8	e_{40}
v_9	x_9	y_9	e_{38}

Table 4.6: Record for Vertex

Face No.	**Bounded Half Edge**
f_0	e_6
f_1	e_8
f_2	e_{36}
f_3	e_{33}
f_4	e_{30}
f_5	e_{28}
f_6	e_{24}
f_7	e_{21}
f_8	e_{17}
f_9	e_{14}
f_{10}	e_{11}
f_{11}	e_{38}

Table 4.7: Record for Face

Algorithm 2: Degree count of a vertex

Input: 'e' be the half edge with the origin at vertex 'v'
Output: total number of degree of a vertex

```
1 start_half_edge ← e
2 degreeCounter = 1
3 while twin(previous(e)) ≠ start_half_edge do
4     e ← twin(previous(e))
5     degreeCount = degreeCount + 1
```

Once the triangulation is available in a dcel data structure, it is very convenient to compute several properties of faces, vertices, and edges. For instance, we can find the degree of a node in time proportional to the value of the degree. This can be briefly explained as follows. We start with the half edge e_i incident at vertex v_j. To go to the next half edge incident at v_j, we follow the twin of the previous of e_i. We can repeat this twin previous sequence to come back to the starting half edge e_i. As we trace the twins of the previous, we can update a counter to determine the number of edges incident at v_j and find the degree. A formal description of the algorithm for determining degree is listed as Algorithm 2. It is straightforward to see that the time for determining the degree by Algorithm 2 is $O(d_i)$ where d_i is the degree of vertex v_j.

We performed several executions of the program to count the number of even/odd degree nodes on the Delaunay triangulation of randomly generated nodes. The number n of randomly generated nodes was taken as n = 50, 100, 200, 300, 400, 500, 600, 700, and 800. For each value of n we generated randomly located nodes five times. The even/odd node degree count values for each five sets of nodes was averaged. To generate a randomly located node nd_i, its x-coordinate and y-coordinate were generated by using the rand function of Java in the range of the size of the canvas. The results are shown in Table 4.8. A snapshot of the count of odd/even degree nodes in the Delaunay triangulation of randomly generated 117 nodes by our program is shown in Figure 4.4 and Figure 4.5.

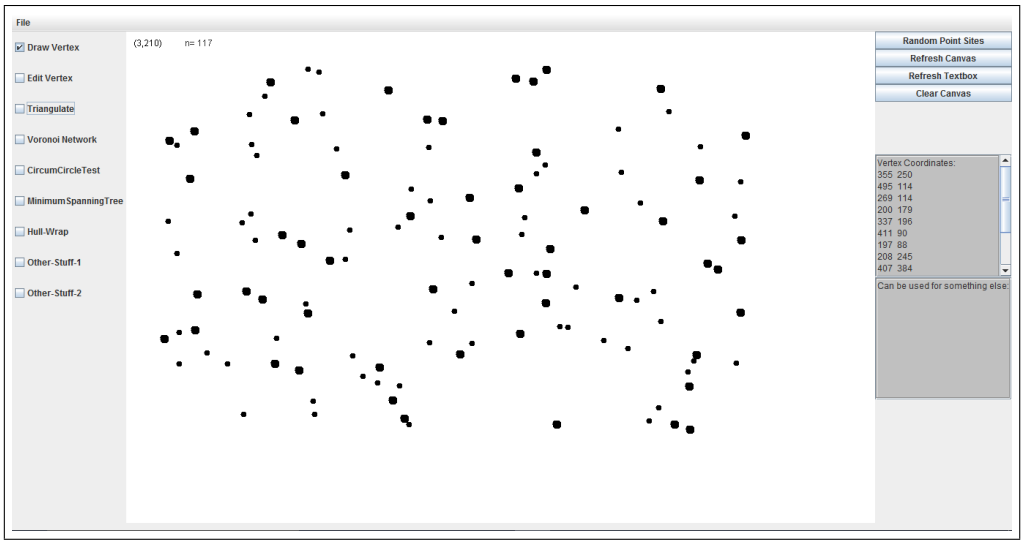


Figure 4.4: Snapshot of 117 randomly generated nodes

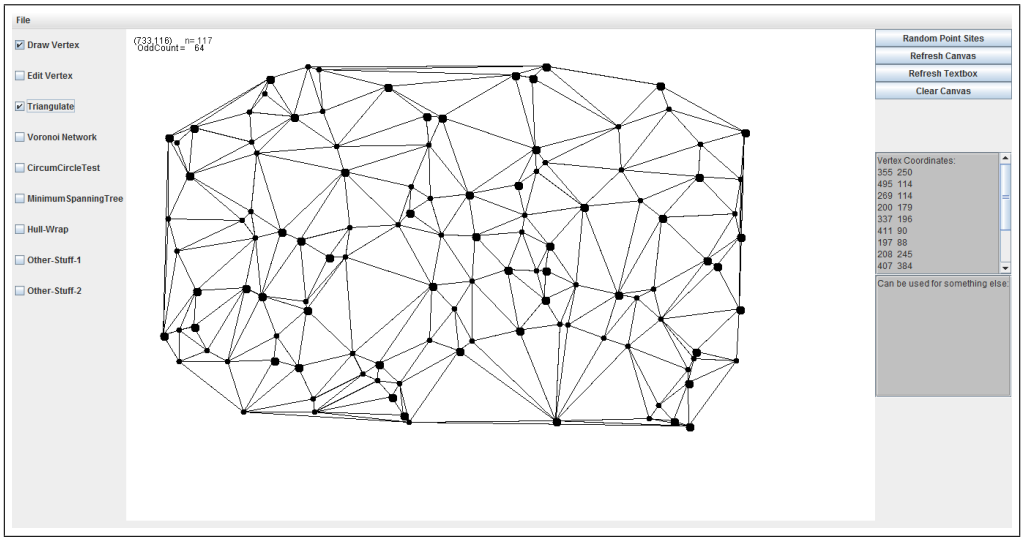


Figure 4.5: Delaunay triangulation of 117 nodes with odd/even count

No. of Nodes	**Odd Degree Count**	**Even Degree Count**	**Odd Percentage**	**Even Percentage**
50	24	26	48.00	52.00
50	28	22	56.00	44.00
50	32	18	64.00	36.00
50	26	24	52.00	48.00
50	30	20	60.00	40.00
Average	28	22	56.00	44.00
100	60	40	60.00	40.00
100	56	44	56.00	44.00
100	44	56	44.00	56.00
100	48	52	48.00	52.00
100	54	46	54.00	46.00
Average	52.4	47.6	52.40	47.60
200	86	114	43.00	57.00
200	116	84	58.00	42.00
200	114	86	57.00	43.00
200	98	102	49.00	51.00
200	104	96	52.00	48.00
Average	103.6	96.4	51.80	48.20
300	164	136	54.67	45.33
300	150	150	50.00	50.00
300	158	142	52.67	47.33
300	148	152	49.33	50.67
300	142	158	47.33	52.67
Average	152.4	147.6	50.80	49.20

Continued on next page

Table 4.8 – continued from previous page

No. of Nodes	Odd Degree Count	Even Degree Count	Odd Percentage	Even Percentage
400	206	194	51.50	48.50
400	192	208	48.00	52.00
400	196	204	49.00	51.00
400	212	188	53.00	47.00
400	204	196	51.00	49.00
Average	202	198	50.50	49.50
500	246	254	49.20	50.80
500	222	278	44.40	55.60
500	242	258	48.40	51.60
500	252	248	50.40	49.60
500	260	240	52.00	48.00
Average	244.4	255.6	48.88	51.12
600	286	314	47.67	52.33
600	296	304	49.33	50.67
600	292	308	48.67	51.33
600	308	292	51.33	48.67
600	312	288	52.00	48.00
Average	298.8	301.2	49.80	50.20
700	364	336	52.00	48.00
700	352	348	50.29	49.71
700	334	366	47.71	52.29
700	376	324	53.71	46.29
700	340	360	48.57	51.43
Average	353.2	346.8	50.46	49.54

Continued on next page

Table 4.8 – continued from previous page

No. of Nodes	**Odd Degree Count**	**Even Degree Count**	**Odd Percentage**	**Even Percentage**
800	378	422	47.25	52.75
800	400	400	50.00	50.00
800	408	392	51.00	49.00
800	414	386	51.75	48.25
800	396	404	49.50	50.50
Average	399.20	346.80	50.46	49.54

Table 4.8: Degree Distribution Count

Chapter 5

Conclusion and Discussion

We presented a cursory review of noteworthy results on node degree aware triangulations of polygonal shapes and point sites in two - dimensional Euclidean planes. Most results on node degree aware triangulation deal with the characterization and development of efficient algorithms. No algorithmic result has been reported that generates a triangulation with the maximum number of even-degree nodes for point sites. It is known that the problem of generating a degree constrained spanning tree is NP-Hard [GJ79]. This means that it may be worth exploring whether the triangulation problem that maximizes the number of even degree points is NP-Hard. Some other constrained triangulations are known to be NP-Hard. As pointed out in Chapter 2, the minimum weight triangulation problem is NP-Hard [MR08]. This further gives us a hint that the triangulation problem that maximizes the total number of even degree nodes is not easy.

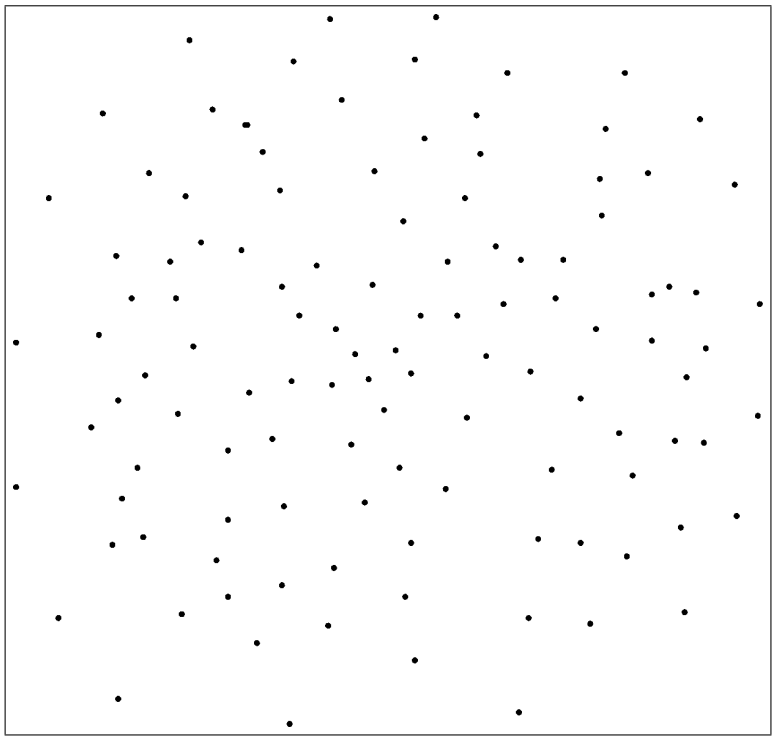
Some structural insight on degree constrained triangulation is needed to settle the problem. The first result we presented was the characterization of node distribution that admits even-degree triangulation. By constructing nested structures as depicted in Figure 2.5 (Chapter 2) we characterized the distribution of point sites that indeed admits even-degree triangulation. The second result we presented was the characterization of the annular region whose triangulation will have the most nodes with odd degree. Specifically, we showed that an annular region with an outer chain with (relatively) a large number of vertices and inner chain just a triangle will have a triangulation of which most vertices are of even degree.

We presented an efficient heuristic algorithm (Alternate Marching Algorithm) that triangulates a normal annular region with the most vertices of even-degree. The time complexity is $O(n^2)$,

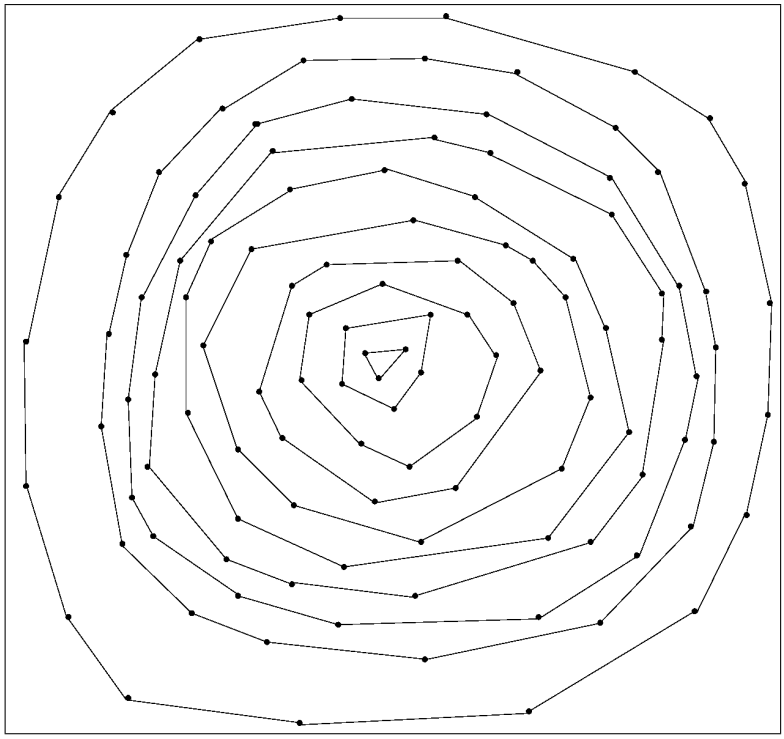
where n is the number of vertices in the region.

We performed an experimental investigation of node degree distribution on the Delaunay triangulation of points generated randomly. The experimental results (Table 4.8) reveal that the average count of even-degree and odd-degree nodes are almost equal - the range is [48% - 50%].

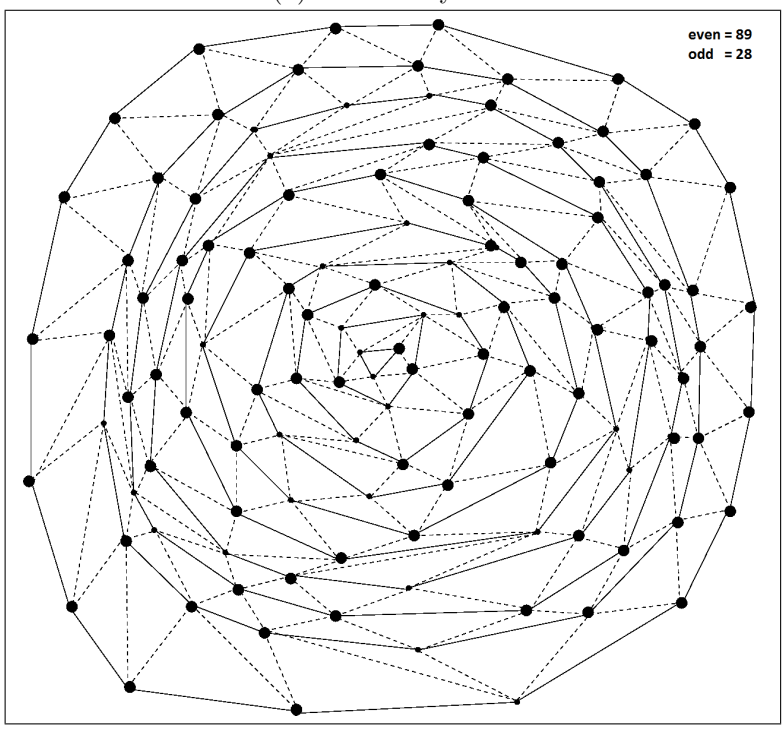
The Alternate Marching Algorithm presented in Chapter 3 can be used to generate triangulations of points in 2D that tend to raise the count of even degree nodes. We propose an approach based on the construction of convex layers. The convex layers induced by a group of nodes are illustrated in Figure 5.1.



(a) Randomly Generated Points



(b) Convex Layers



(c)

Figure 5.1: Alternate Marching Approach

A structure closely related to a spiral chain using the convex layers of point sites is shown in Figure 5.1. Figure 5.1a is a randomly generated distribution of 117 point sites. Figure 5.1b is the convex layers for this point distribution. Convex layers of point sites have been investigated in the computational geometry literature [O'R98]. The outermost layer (first layer) are the points on the convex hull. If we remove the point sites on the first layer and recompute the convex hull then the points on the new convex hull give the second convex layer and so on. The point sites distribution shown in Figure 5.1a will have 10 convex layers as shown in Figure 5.1b. A triangulation of these convex layers by using the alternate marching algorithm produces nodes with a very high percentage of even degree. In the example of Figure 5.1c, 76.07% of the nodes (89/117) are of even degree.

It would be interesting to perform more experiments to evaluate the performance of the alternate marching algorithm on various randomly generated point sites.

Bibliography

[AH96] Thomas Auer and Martin Held. Heuristics for the generation of random polygons. In *Proceedings of the 28th Canadian Conference on Computational Geometry, CCCG 2016, August 12-15, 1996, Carleton University, Ottawa, Canada*, pages 38–43, 1996.

[AHH+09] Oswin Aichholzer, Thomas Hackl, Michael Hoffmann, Alexander Pilz, Günter Rote, Bettina Speckmann, and Birgit Vogtenhuber. Plane graphs with parity constraints. In *Algorithms and Data Structures, 11th International Symposium, WADS 2009, Banff, Canada, August 21-23, 2009. Proceedings*, pages 13–24, 2009.

[AHMS96] Esther M Arkin, Martin Held, Joseph SB Mitchell, and Steven S Skiena. Hamiltonian triangulations for fast rendering. *The Visual Computer*, 12(9):429–444, 1996.

[BEE+93] Marshall Bern, Herbert Edelsbrunner, David Eppstein, Scott Mitchell, and Tiow Seng Tan. Edge insertion for optimal triangulations. *Discrete & Computational Geometry*, 10(1):47–65, 1993.

[BLMV16] Therese C. Biedl, Anna Lubiw, Saeed Mehrabi, and Sander Verdonschot. Rectangle-of-influence triangulations. In *Proceedings of the 28th Canadian Conference on Computational Geometry, CCCG 2016, August 3-5, 2016, Simon Fraser University, Vancouver, British Columbia, Canada*, pages 237–243, 2016.

[BMS94] Christoph Burnikel, Kurt Mehlhorn, and Stefan Schirra. On degeneracy in geometric computations. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '94, pages 16–23, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[Cha91] Bernard Chazelle. Triangulating a simple polygon in linear time. *Discrete & Computational Geometry*, 6(3):485–524, 1991.

[dBvKOS97] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[For87] Steven Fortune. A sweepline algorithm for voronoi diagrams. *Algorithmica*, 2:153–174, 1987.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[GKS92] Leonidas J. Guibas, Donald E. Knuth, and Micha Sharir. Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 7(4):381–413, 1992.

[GM91] Subir Kumar Ghosh and David M. Mount. An output-sensitive algorithm for computing visibility graphs. *SIAM Journal on Computing*, 20(5):888–910, 1991.

[GS85] Leonidas Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computations of voronoi diagrams. *ACM Tmns. Graph*, pages 74–123, 1985.

[Gya12] Roshan Gyawali. Degree constrained triangulation. Master's thesis, University of Nevada, Las Vegas, 2012. An optional note.

[HK96] Frank Hoffmann and Klaus Kriegel. A graph-coloring result and its consequences for polygon-guarding problems. *SIAM J. Discret. Math.*, 9(2):210–224, May 1996.

[HM83] Stefan Hertel and Kurt Mehlhorn. Fast triangulation of simple polygons. In *Proceedings of the 1983 International FCT-Conference on Fundamentals of Computation Theory*, pages 207–218, London, UK, UK, 1983. Springer-Verlag.

[HN96] Ferran Hurtado and Marc Noy. The graph of triangulations of a convex polygon. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry*, SCG '96, pages 407–408, New York, NY, USA, 1996. ACM.

[HNU96] F. Hurtado, M. Noy, and J. Urrutia. Flipping edges in triangulations. In *Proceedings of the Twelfth Annual Symposium on Computational Geometry*, SCG '96, pages 214–223, New York, NY, USA, 1996. ACM.

[Hut04] D.V. Hutton. *Fundamentals of Finite Element Analysis*. McGraw-Hill series in mechanical engineering. McGraw-Hill, 2004.

[Law77] C. L. Lawson. Software for c1 surface interpolation. *Mathematical Software III*, pages 161–194, 1977.

[Mei75] G. H. Meisters. Polygon have ears. *The American Mathematical Monthly*, 82:648–651, 1975.

[MP78] David E. Muller and Franco P. Preparata. Finding the intersection of two convex polyhedra. *Theor. Comput. Sci.*, 7:217–236, 1978.

[MR08] Wolfgang Mulzer and Günter Rote. Minimum-weight triangulation is np-hard. *J. ACM*, 55(2):11:1–11:29, May 2008.

[OB08] Eliyahu Osherovich and Alfred M. Bruckstein. All triangulations are reachable via sequences of edge-flips: an elementary proof. *Computer Aided Geometric Design*, 25:157–161, 2008.

[O'R87] Joseph O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, Inc., New York, NY, USA, 1987.

[O'R98] Joseph O'Rourke. *Computational Geometry in C.* Cambridge University Press, New York, NY, USA, 2nd edition, 1998.

[PRVU10] Canek Peláez, Adriana Ramırez-Vigueras, and Jorge Urrutia. Triangulations with many points of even degree. In *Proceedings of the 22nd Annual Canadian Conference on Computational Geometry*, pages 103–106, 2010.

Curriculum Vitae

Graduate College

University of Nevada, Las Vegas

Bhaikaji Gurung

Degrees:

Bachelor of Engineering in Computer Engineering 2010

Tribhuwan University, Pulchowk Campus, Nepal

Thesis Title: Degree constrained triangulation of annular region and point sites

Thesis Examination Committee:

Chairperson, Dr. Laxmi Gewali, Ph.D.

Committee Member, Dr. John Minor, Ph.D.

Committee Member, Dr. Justin Zhan, Ph.D.

Graduate Faculty Representative, Dr. Henry Selvaraj, Ph.D.