M.Sc. Engg. Thesis

Dynamic Maintenance of View of Points in Three Dimensional Euclidean Space

by
Mohammad Abdul Wahid

Submitted to

Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000

June, 2011

The thesis titled "**Dynamic Maintenance of View of Points in Three Dimensional Euclidean Space**", submitted by Mohammad Abdul Wahid, Roll No. 100705029F, Session October 2007, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on June 11, 2011.

Board of Examiners

Candidate's Declaration

This is to certify that the work entitled "Dynamic Maintenance of View of Points in Three Dimensional Euclidean Space" is the outcome of the research carried out by me under the supervision of Dr. M. Kaykobad in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka-1000. It is also declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Mohammad Abdul Wahid
Candidate

Contents

List of Figures

Acknowledgments

All praises are due to Allah, the most benevolent and merciful.

I express my heart-felt gratitude to my supervisor, Professor Dr. M. Kaykobad for his constant supervision of this work. He helped me a lot in every aspect of this work and guided me with proper directions whenever I sought one. His patient hearing of my ideas, critical analysis of my observations and detecting flaws (and amending thereby) in my thinking and writing have made this thesis a success.

I want to show my gratitude to Professor Dr. Md. Monirul Islam for his patience, valuable suggestions, comments and important corrections. I would like to thank Dr. Masud Hasan for his constant support during this research as well as for his guidance through the early years of chaos and confusion. I am also thankful to the external member of the board of examiner of my thesis, Dr. Shamim Ahmad for his valuable suggestions and comments.

I remain ever grateful to my beloved parents, brothers, sisters and all of my teachers from primary education to postgraduate education who always exist as sources of inspiration behind every success of mine I have ever made. Finally, I can not forget my friends, seniors and juniors specially Mr. Md. Zahid Hossain, Mr. Mahmudul Faisal Al Ameen, Mr. Mohammad Monoar Hossain, Mr. Sanjib Kumar Saha, Mr. Md. Amirul Islam and others for their encouragement and support during my thesis.

Abstract

A continuous change of relative positions of a set of points is perceived by a human observer only discretely. The reflection of a significant change in visibility in 3-dimensional Euclidean space in the perceived picture is called the change of *view*. The objective of this research is to define the term *view* and to provide necessary data structures and algorithms to maintain it dynamically. To determine the mentioned significant change in visibility, it is important to detect the presence of empty circle. So, to get the efficient query time for detecting the emptiness of a circle, we reveal some important properties of Gabriel graph. The main achievements of this thesis are a suitable data structure and efficient algorithms for maintaining the *view* dynamically with responsiveness $O(M \log M)$ and efficiency $O(M \log{(M)} \lambda_s(n^2))$ where M is the degree of the moving vertex in the Delaunay graph and $\lambda_s(n)$ is the maximum length Davenport-Schinzel sequence of n symbols with order s. In fact, the growth of $M \log M$ is very low because M is an amortized constant.

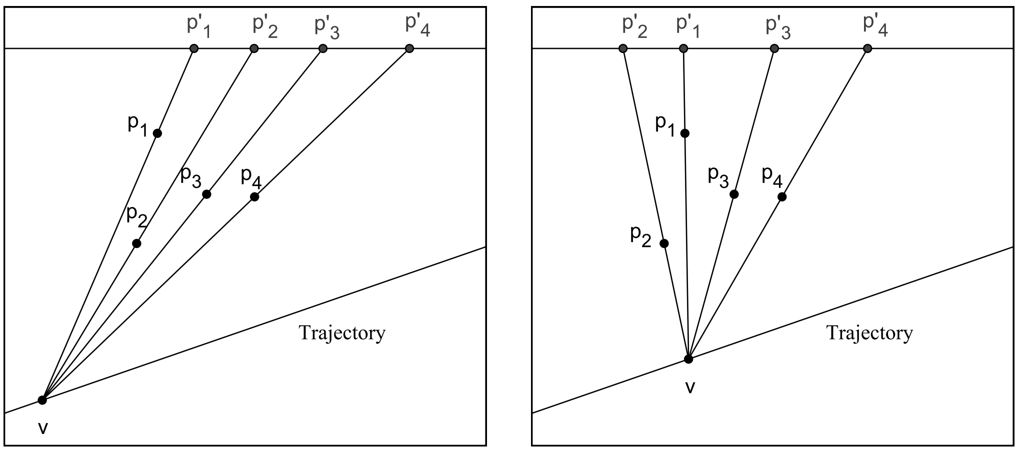
Chapter 1

Introduction

1.1 Introduction

Let P be a set of points in two dimensional Euclidean space, $\mathbb{R}^2$. According to Devillers [1], a point $p \in P$ is visible from $v \in \mathbb{R}^2$ if the line segment pv contains no other point of P. The *view* of P from a view point v is the clockwise circular ordering of all points in P that are visible from v [Figure 1.1]. To maintain the circularly ordered set of points around v, one of the basic idea is to sort them first (as a pre-process), then, for each of the n pairs of circularly consecutive points, insert an event in an event-queue that tells when the swapping of two consecutive points should happen (if ever). The update is as follows: (a) Pop an event. (b) Update the circular ordering by swapping two consecutive elements. (c) Insert up to three new events in the queue, while deleting two old events. However, there is no complete definition of *view* for 3D point set based on the order of points as well as in the perspective of human observation.

If P is the set of points in three dimensional Euclidean space, $\mathbb{R}^3$ and P' be the set of projected points taken by the ray shooting from the view point $v \in \mathbb{R}^3$ to P then for the movement of the view point v, each $p \in P'$ changes its position on the projection plane. More specifically, if the view point moves from left to right along a straight line trajectory, the projected points will move from right to left on the projection plane. So, total two events can be inserted: one event



(a) The order is: p'_1, p'_2, p'_3, p'_4 (b) p'_1 and p'_2 changes their relative order

Figure 1.1: Changing the order of points in the projection line gives new view in 2D

in between two consecutive points determined by x-coordinate and another event in between two consecutive points determined by y-coordinate in the projection plane. Therefore, to maintain the order among projected points, the configuration of each *view* can be kept before the crossing of two points either in x-coordinate or in y-coordinate. To accomplish this task, a spatial subdivision of the projection plane has been done based on each of the four neighbouring points. Thus, a bounded rectangular region can be defined as the *safe region* for each point. The term *safe region* suggests that in spite of the movement of a point $p \in P'$, if p remains in its own safe region then no topological event will be occurred. The safe region for each point in the projection plane has been maintained through a Data Structure called *Neighbourhood Graph (NG)*. The necessary algorithm for NG has also been given for updating the certificate for each topological events.

If two points change their neighbourhood relationship occupying an empty circular area considering these two points as the diameter, natural view get affected and a significant change is felt. So, a naive algorithm has been provided for testing the emptiness of any circle. Furthermore, with the help of

Davenport-Schinzel sequences it is shown that the proposed NG has the efficiency $O(k\lambda_s(n^2))$, locality $O(1)$ and responsiveness $O(k)$. Here, k is the number of test for checking whether the circle is empty or not. Both the efficiency and responsiveness of the algorithm for maintaining the *view* become costly when $k = n - 2$. However, in such case, the value of k is much less than $n - 2$ for all other pairs of points. This motivate us for further research on the configuration of empty circles in a point set. Rather than testing for the emptiness of a circle by a repeated checking, it is possible to maintain a graph named empty circle graph such. There is an edge p_ip_j in the empty circle graph if the circle with diameter p_ip_j is empty. Throughout the paper, the circle p_ip_j refers to the circle with diameter p_ip_j. Therefore, the emptiness of a circle can be determined very quickly which in fact gives us an efficient response time to maintain the *view*. However, the major challenges here is to update the empty circle graph for moving point set.

Let us consider a Delaunay graph DG constructed from the point set P' and p_j is the Delaunay neighbor of p_i in DG. The perpendicular line to p_ip_j define the boundary of two half planes. The half plane inscribed p_i can be called as the left half plane of p_i and the opposite half plane that inscribe p_j can be called as the right half plane of p_i. For all p_j incident to p_i, a set of left half planes can be found and the intersection of all these left half planes of p_i give a convex region called effective region $R(p_i)$. It is shown in the thesis that if a new point p_k is inserted anywhere in $R(p_i)$ then p_k will be the Gabriel adjacent to p_i and hence, the circle p_ip_k will be empty. The intersection of $R(p_i)$ for all $p_i \in DG$, another set of cells named *effective cells* can be obtained. Each *effective cell* has such an important property that if a point p_k does not leave out from the *effective cell* then the set of Gabriel adjacent of p_k remains constant.

Gabriel graph is the subgraph of Delaunay graph and Delaunay edges are responsible for constructing the *effective regions* for Gabriel graph. So, for a dynamic scenario, it is necessary to update all the effective regions as well as the

Delaunay graph efficiently. It is shown in this paper that the cost for updating the Delaunay graph from the Gabriel graph is $O(M \log(M))$ where M is the degree of Delaunay graph. Along with the maintenance of *view*, this results an empty circle test with $O(M \log(M))$ responsiveness and $O(M \log(M)\lambda_s(n^2))$ efficiency. It has also been shown that the growth of $M \log(M)$ is very low because M is in fact amortized constant. Beside these, the expected degree of M is only 6 and in practical case the value of M rarely exceed 16. As a byproduct of the proposed spatial subdivision of the Gabriel graph, a large number of applications have been proposed in various research fields like gene sampling, clustering, wireless sensor network etc.

1.2 Motivation

This problem is motivated by the graphics problem of maintaining the *view* dynamically during an interactive walk through a 3D scene. The term frame-to-frame coherence refers to the similarity between consecutive frames in an animation. This similarity makes it wasteful to render each frame from scratch. Since Z-buffer algorithm needs huge amount of memory for an computer generated animation the Binary Space Partitioning could be an appropriate approach which run once for a static scene and produces a tree of size $\Omega(n)$ and $O(n^2)$, where n is the size of the scene. For each rendered image, a painter type algorithm traverses the tree. However, in the real-time algorithm, the cost of traversal is prohibitive even if the size of the tree is linear in the size of the input.

The idea of maintenance of object ordering in 2D by Devillers [1] also motivates us to provide the possible way of such object ordering for 3D point set. Additionally, we consider our *view* point is more likely a human observer.

1.3 Related Works

Several researchers [1, 2, 3] presented a faster solution of moving view point query problem among points in 2D with provable worst-case and amortized complexity. Devillers [1] reported that solving the problem in 3D remains a difficult open problem. Bern [4] also investigated 3D visibility problems in which the viewing position moves along a straight flight path and have developed algorithm for discovering topology changes. Akbari [5] works with the maintenance of visibility of a moving segment observer inside the polygon with holes and raises three issues related to visibility maintenance: basic problems, query problems and kinetic problems. Providing appropriate data structure for the incremental update of a dynamic scenario is a big challenge in computational geometry. The data structure that is specially designed to update a dynamic scenario is called *Kinetic Data Structure (KDS)* [6, 7]. Harry Plantinga [8] used aspect graph for dynamically maintaining the object topology of the image of a polyhedron changes with changing viewpoint.

KDS do not pose major implementation problems and perform well on several natural point distribution [9]. The average case behavior of some discrete attributes like closest pair, convex hull etc., can be defined on points moving on random trajectories. For the random distribution on a unit square, the voronoi diagram changes $\theta(n^{1+\frac{1}{d}})$ times and the closest pair changes $\theta(n^{\frac{2}{d}})$ times [10]. The planar subdivision of the free space between two polygons called *external relative geodesic triangulation* are proposed as KDS to maintain collision detection between two simple polygons [11].

Getting an efficient solution for determining whether a circle is empty or not, empty circle graph (Gabriel graph) has been studied thoroughly. In 1969, Gabriel and Sokal [12] in their paper on Geographic variation analysis used empty circle graph for geographic connectivity. Since then, the graph is also known as Gabriel graph. The graph is later studied extensively in several research areas. Choo [13] proposed a agglomerative clustering algorithm called MOSAIC which greedily merges the neighbouring clusters based on the Gabriel neighbourhood. Many researchers proposed the applications of Gabriel graph for the exploratory analysis of potentially high dimensional labeled data [14, 15]. A comprehensive study on the relationship between Support Vector Machine (SVM) and Gabriel graph is provided by Zhang [16]. They showed that based on the Gabriel graph's training data set reduction algorithm, it is possible to improve the performance of SVM. Moreover, support vectors are actually the subset of Gabriel edited set. Gabriel graph is also widely used in geographical genetic structure, genetic relation and gene flow analysis [17]. Since there are spatial trends for genetic variation, the adjacency relationship of Gabriel graph provides significant information about genetic distance and gene flow analysis.

Due to the wide range of applications of Gabriel graph, Matula and Sokal [18] revealed some important properties of Gabriel graph for geographic variation research and clustering of point in the plane. Howe [19] proved that Gabriel graph is the subgraph of the Delaunay graph and the construction of the Gabriel graph from the Delaunay graph can be done $O(n)$ times. A lot of works have also been done on the spanning ratio of Gabriel graph [20], higher order Gabriel graph and Gabriel graph for higher dimension [21, 22] etc. In spite of having huge number of applications of Gabriel graph, no effort has been given to provide for getting the spatial subdivision of the domain of Gabriel graph.

1.4 Outline of this thesis

The whole research work is organized mainly into the following Chapters:

Chapter 2: The Chapter is intended for getting the necessary preliminary ideas before going to the original discussion. The idea of dynamic maintenance, associated framework for measuring the performance of a dynamic update, relationship between combinatorial structure and Davenport-Schinzel sequence are introduced in the first few sections. In the last section in this Chapter, the basic idea for maintaining the Delaunay graph is given that is necessary for us to compare with the algorithm provided for dynamic update of Gabriel graph in Chapter 5.

Chapter 3: A naive solution for maintaining the *view* dynamically is provided in this Chapter. After providing the definition of *view* in the first section, a naive algorithm with necessary data structure has been designed in section 2. The last section deal with integrating the empty circle test and a proof scheme for measuring the performance.

Chapter 4: Motivated by the empty circle test, a thorough study has been done on empty circle graph (Gabriel graph) in the whole Chapter. We reveal various important properties of Gabriel graph and show its lower and upper bound in the number of edges in it.

Chapter 5: To minimize the cost of empty circle test, we propose a technique named spatial subdivision of the space of Gabriel graph. The first few sections are dedicated for describing the construction techniques of *sparse cell*, *effective region*, *effective cell* and its associated idea. In the final section, we provide the algorithm for the dynamic update of the *effective regions* of each point efficiently.

Chapter 6: The results and a discussion on the outcome of this thesis is described in this Chapter.

Chapter 7: This Chapter draws the conclusion providing the future directions for further study related to this thesis.

Chapter 2

Preliminaries

2.1 Dynamic Maintenance

Dealing with the real world simulation by computer raises the issues of discrete and continuous aspects. Consider a geometric objects in motion for example, the convex hull of a set of n points in a plane. As the point move continuously, the convex hull is not changed continuously rather it changes after a certain discrete moments. We may also interested to know the closest distance between all pairs of objects in a set. Each distance is certainly a real number and change continuously in each continuous movement of objects. However, the pair with closest pair is changed after a discrete period of time. That is why, we can say the closest pair is a discrete attribute. To get the discrete moment of changing the combinatorial structure (attribute) like convex hull, closest pair, Delaunay graph and so on, we can assign some geometric relations which certify the attribute. All we need to find the moment when one of the relations become invalid. Once it is invalid, we say, the combinatorial structure is changed. So, there are two major tasks related to this issue. One is getting the appropriate geometric relations for certifying the attribute and another is updating each relation if at least one of them become invalid. The update technique for maintaining any discrete attribute of a dynamic configuration is also known as Dynamic Maintenance.

2.2 Framework

Here we adopt the framework from [23] which is essential for analysis the performance of Kinetic Data Structure of mobile data. Given a set S of items, a *configuration* π associates with each item the *position* of a point in the projection plane. $s(\pi)$ denotes the position of an item s under the configuration π. We can also use only s instead of $s(\pi)$ in the context where the configuration π is clearly defined.

Definition 1. *A* Kinetic Data Structure(KDS) *is a type of data structure that maintains a certain geometric attribute using a collection of simple geometric relations that certifies the combinatorial structure of the attribute in any configuration. It also provides a set of rules for repairing those certifying relations for maintaining the attribute when one of the relations fails. The process of maintaining the KDS accordingly is called* Kinetisation.

Definition 2. *An* Attribute *is a function that associates with each configuration π a combinatorial structure based on S. For an attribute A, we denote by $A(\pi)$ its value at configuration π. For example, the closest pair, Voronoi Diagram, convex hull and our defined* view *all are taken as the function of configuration. So, all are the attributes of the configuration π.*

Definition 3. *A* Certificate $C : A^m \to \{-1, 0, 1\}$ *acting on m-tuple of points is simple geometric relations associated a real number with each configuration of these items. When this real number is positive for a given π, the certificate is said to be* valid. *When it is negative, we say it is* invalid *and when zero we say* degenerate. *We write $[a < b]$ for the certificate that associates with a configuration π the quantity $b(\pi) - a(\pi)$.*

Definition 4. *A* Topological Event *is the failure of a certificate during motion of sites in a configuration. The future time of failure of a certificate can be predicted if the motion plan for the all the associated sites are beforehand known. There are* *two types of events: external and internal. If the combinatorial structure of the attribute changes then we call it internal event. On the other hand if any certificate needed to be updated although combinatorial structure of the sites remain same.*

Example 1. *For instance, our task is to maintain the right most point in a line. An event occurs when two points change their order. However, attribute will be changed only when the right most point change its order. The former case is internal event and the later is called external event.*

2.3 Proof Scheme

A *Proof Scheme* for an Attribute A associates a set of certificates C with each configuration in general position. In computational geometry, it is assumed that a configuration is in general position so that none of the geometric test or certificate used is degenerate in this configuration. The performance of a KDS is measured by four criteria:

- The *locality* of the proof scheme is the worst case number of certificates any given item is involved in. In KDS, we add a data structure to a proof to help in the proof update when a certificate fails.
- The *responsiveness* is the response time of updating a certificate in a KDS i.e. the worst case computational cost of processing a complete event.
- The *Compactness* of a KDS is the total number of certificates that are needed to be stored. So, we say a KDS is *compact* if the size of certificate set it needs is close to linear in the degrees of freedom of any configuration.
- *Efficiency* deals with the number of events need to be processed. A KDS is *efficient* if the total number of events it needs to process is comparable to the number of required changes in the external events. So, efficiency can

also be defined as the ratio of the total worst case number of internal and external event to the worst case number of external events.

2.4 Davenport-Schinzel Sequence

Davenport-Schinzel (DS) sequences are very interesting and powerful combinatorial structures that arise in the analysis of lower or upper envelope of collection of univariate function. DS-sequences play a very important role on finding the complexity of geometric and algorithmic problem related to arrangement of curves and surfaces.

Definition 5 ([24]). *Let n and s be two positive integers. A sequence $U = u_1, u_2, u_3, \ldots, u_m$ of integers is an $(n, s) Davenport-Schinzel$ sequence ($DS(n, s)$-sequence for short) if $u_i \neq u_{i+1}$ for $i < m$ where $1 \leq u_i \leq n$ for each $i \leq m$ and there does not exist $s + 2$ indices $1 \leq i_1 < i_2 < \cdots < i_{s+2} < m$ such that $u_{i_1} = u_{i_3} = u_{i_5} = \cdots = a$, and $u_{i_2} = u_{i_4} = u_{i_6} = \cdots = b$ and $a \neq b$.*

In the definition 5, the last condition forbids the presence of long alternation of any pair of distinct symbols. We say s as the order of U to n as the number of symbols composing U, $|U|$ is the length m of $DS(n, s)$.

$$\lambda_s(n) = max\{|U| \mid U\ is\ a\ DS(n, s) - sequence\}. \tag{2.4.1}$$

Definition 6. *Let $F = \{f_1, f_2, f_3, \ldots, f_n\}$ be a collection of n real-valued, continuous totally defined functions so that the graphs of every pair of distinct functions intersect in at most s points. the lower envelope of F is defined as $E_{F(x)} = min_{1 \leq i \leq n} f_i(x)$.*

In Definition 6, E_F is the point-wise minimum of $f_i \in F$. Let $I_1, I_2, I_3, \ldots I_m$ be the intervals on the x-axis so that they cover the entire x-axis.

Corollary 1 ([24]). *For any collection $F = \{f_1, f_2, f_3, \ldots, f_n\}$ of n continuous, totally defined, univariate functions, each pair of whose graphs intersect in at most*

s points, the length of the lower-envelope sequence $U(F)$ is at most $\lambda_s(n)$, and this bound can be attained by such a collection F.

It is shown by Sharir et al. [24] that $\lambda_1(n) = n$, $\lambda_2(n) = 2n - 1$ and $\lambda_3(n) = \Theta(n\alpha(n))$ where $\alpha(n)$ =inverse Ackermann function [25]. Later Agarwal et. al [26] and Sharir et al. [27] proved that $\lambda_s(n)$ is nearly linear in n for any fixed $s > 3$.

Inverse Ackermann Function

There are many modified versions of Ackermann function to suit various purposes. One common version, the two-argument Ackermann-Peter function, is defined as follows for non negative integers m and n: [28]

$$A(m,n) = \begin{cases} n+1 & \text{if } m = 0 \\ A(m-1,1) & \text{if } m > 0 \text{ and } n = 0 \\ A(m-1, A(m, n-1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

Although the value of $A(1,2)$ is as low as 4, the value of $A(4,3)$ is surprisingly high.

$$\begin{aligned} A(1,2) &= A(0, A(1,1)) \\ &= A(0, A(0, A(1,0))) \\ &= A(0, A(0, A(0,1))) \\ &= A(0, A(0,2)) \\ &= A(0,3) \\ &= 4. \end{aligned}$$

$$\begin{aligned} A(4,3) &= A(3, A(4,2)) \\ &= 2^{2^{65536}} - 3. \end{aligned}$$

If the value of $A(4,3)$ is written as a power of 10, this is roughly equivalent to $10^{6.031\times 10^{19727}}$. Let the function $f(n)$ be an Ackermann function of $A(n,n)$ then $f(n)$ grows very rapidly. The inverse function of Ackermann function f^1 usually denoted by $\alpha(n)$, grows very slowly. In fact, $\alpha(n)$ is less than 5 for any practical input size n.

2.5 Dynamic Update of Delaunay Graph

Due to the wide range of application, Delaunay graph is one of the most studied graph in computational geometry. The Dynamic update for insertion and deletion of a point in the graph require $O(n)$ running time in worst case. One of the best known approach to solve this problem is provided by Kao et. al [29] and Guibas et al [30] which costs amortized $O(\log n)$ time. Guibas use history graph graph to keep the track of all previous update. If a point is inserted into the triangle $\triangle\ abc$ then it is replaced by the new three triangles. In history graph, $\triangle\ abc$ remains as part of the data structure marking as “old” and pointer are added from $\triangle\ abc$ to each of the newly created triangle as its children. The number of children of $\triangle\ abc$ is either three or two. Starting with a unit square as a single node, the diagonal produces two children nodes of the unit square. Now, the incremental insertion of each points in any triangle extend the history graph with a hierarchical relationship. To find the location of a triangle containing newly inserted point, we can start from the root node to the finding nodes by tracing the chronological order of old triangles containing the inserted point. Since each old triangles contain at most three new triangles at the next level, it takes constant time to determine appropriate triangle in the next levels containing the query point. Guibas et al. [30] shows that the number of triangles where a point moves, when averaged over all points and all insertions, is only $O(\log n)$ in the expected case. A good advantage of history graph is its persistence. Since it preserves the history of insertion and deletion, it is possible to reverse the all of the previous actions taken into the graph. Therefore, deletion can be done efficiently. However, the main disadvantage of the history graph is the growth of its height is very rapid and hence the point location query takes a large amount of times.

To solve this problem, Kao et al. [29] re-balance the history graph periodically by reconstructing the history graph from the scratch for the current set of active points and destroy the old graph. The operation is called reorganization which is described briefly: if t is the total number of performed insertion or deletion operations and $n(t)$ is the number of active points in the triangulation after the $t-th$ request. Thus $n(0) = 4$. Let t_0 is the time (i.e. request number) of last reorganization. After $t-th$ operation the reorganization is done if $t - t_0 > n(t)$. If s is the request number between t and t_0 and n_0 be the number of active points after the latest reorganization then the expected size of the history graph after the $s-th$ request, $E(s) = n_0 + (s - t_0)$. Again, since, at most one point can be lost from active points per insertion/deletion request, so we have:

$$n_0 \leq n(s) + (s - t_0) \tag{2.5.1}$$

$$n_0 + (s - t_0) \leq n(s) + 2(s - t_0) \tag{2.5.2}$$

$$E(s) \leq n(s) + 2n(s) \tag{2.5.3}$$

$$\leq 3n(s). \tag{2.5.4}$$

So, the maximum size of the history graph is not larger than the number of active point at any instant of time. So the cost of searching the graph does not increase asymptotically and it is $O(\log{(n_0 + s - t_0)}) = O(\log n(s))$. One problem is the reconstruction of the history graph takes itself $O(n \log n)$ times. However, it will not dominate the overall cost if the number of insertions/deletions from the last reorganization is at least as large as $n = n(t)$. Since, to perform reorganization it must be $t - t_0 > n(t)$, the reorganization is not very frequent to increase the overall running time of the algorithm. Thus, the algorithm obtain amortize $O(\log n)$ expected running time.

Chapter 3

Maintenance of View

3.1 Definition

Definition 7. *Let P be a set of points in three dimensional Euclidean space, $\mathbb{R}^3$. The ray shooting from the view point $v \in \mathbb{R}^3$ gives a set of points, $P' = \{p_1, p_2, p_3, \dots\}$ in the projection plane $\mathbb{R}^2$. Then before and after the topological event (i.e. the moment of crossing of two consecutive points $(p_i, p_j) \in P'$ with the condition that the circle with diameter p_ip_j contains no other points in P') are defined as two different views.*

3.2 Spatial Subdivision

For each point $p_i \in P'$, we define a rectangular region as p_i's own safe region because no topological events will occur if p_i does not go out from its own safe region.

Let $p_i.left,\ p_i.right,\ p_i.up,\ p_i.down$ are the the nearest left, nearest right, nearest up and nearest down points respectively of p_i. Therefore, the four corners of the rectangular safe region (R_i) of p_i is determined by these nearest four points (left, right, up and down). To goes out from R_i, p_i will have to cross one of the four boundaries from $p_i.left.x$, $p_i.right.x$, $p_i.up.y$, $p_i.down.y$. So, we have four certificates for each of the points $p_i \in P'$ which will maintain a particular *view*:

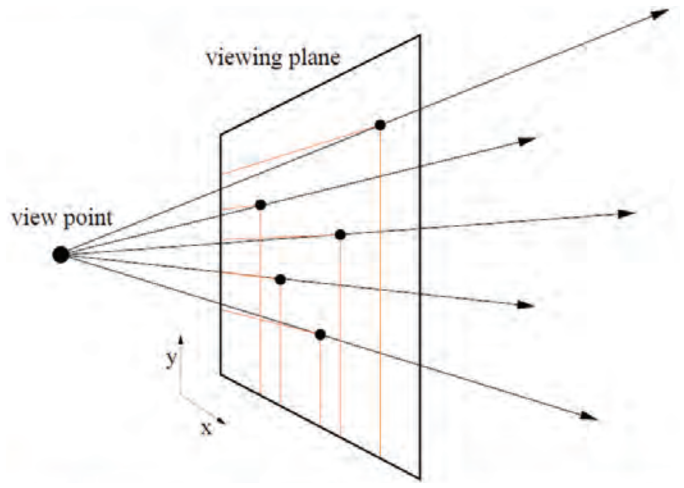


Figure 3.1: View in 3D

$C = \{[p_i >_x p_i.left], [p_i <_x p_i.right], [p_i >_y p_i.down], [p_i <_y p_i.up]\}$.

Here, in the notations $>_x$ and $>_y$, subscript x and y are used to compare the order between two points by x-coordinate and y-coordinate respectively.

If we construct the safe region for all points in P', there will be found a global rectangular region which will inscribe all the points in P'. The leftmost, rightmost, uppermost and lowermost four points can be called as the extreme points in P' because left extreme point do not have any left boundary (other three extreme points have the similar properties). So it can move towards left without violating any certificates. The bounded rectangular region made these four extreme points is called the *global region*.

We consider a point is in *general position* by assuming that no two points change their relative order both in x-axis and y-axis at the same time, i.e. no collision of points will occur as well as no two pairs cross each other either in x-axis or in y-axis at the same time. One more assumption is that the view point will not enter into the convex polyhedra of the point set P. Since, we are mapping our problem domain from $P \in \mathbb{R}^3$ into $P' \in \mathbb{R}^2$, we call P' as $P \in \mathbb{R}^2$ in rest of the parts of this paper.

3.3 Dynamic Maintenance

3.3.1 Neighbourhood Graph

Neighbourhood Graph (NG) is a directed graph containing parallel/multiple edge. In NG, each edge $e_{ij}.\zeta_k \in E(NG)$ has a type ζ_k where $1 \leq k \leq 4$ and $\zeta = \{left, right, up, down\}$. Here, NG preserve the neighbourhood relationship of item i and j through k.

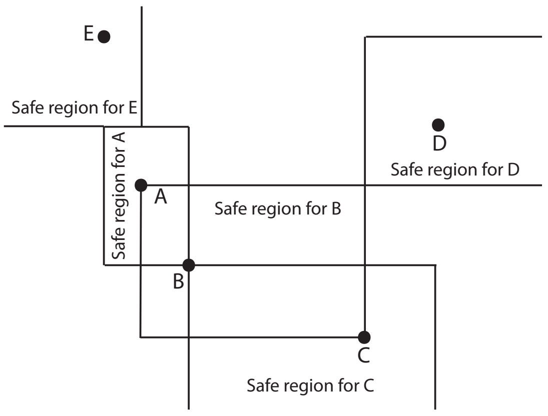
For our problem, each vertex v_i in NG represent the point $p_i \in P$ and $v_i.\zeta_k$ indicates another vertex which is connected by the edge $e_{ij}.\zeta_k \in E(NG)$. An edge $e_{ij}.\zeta_k \in E(NG)$ exists between two vertex v_i and $v_j \in V(NG)$ if the point p_j is one of the four neighbour of the point p_i of type ζ_k. From the attribute we can say that $Indegree(v_i) = Outdegree(v_i) \leq 4$. for any $1 \leq k \leq 4$, if edge $\forall_{1 \leq j \leq n, i \neq j}\{e_{ij}.\zeta_k\}$ is missing then p_i is one of the exterior point in the configuration π. To construct NG, we can use *Linked List* with four pointer and one data section per node where each of the pointer will point its neighbouring point to connect. ∞ is used in pointer ζ_k if there is no edge type ζ_k for a point.

3.3.2 Maintenance Algorithm

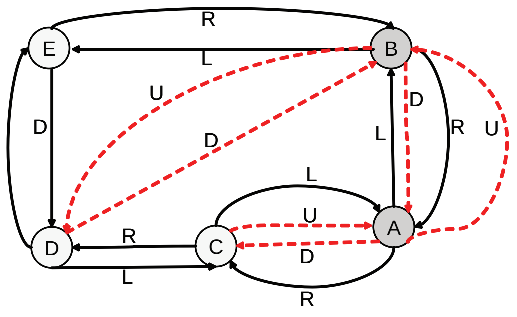
Each certificate is stamped with its failure time and placed into the global event queue. At the time of processing of the first event in the event queue, we need to update the certificate list. Lets a scenario where two sites A and B participate in violating each certificates:

Event: Failure of Certificate $[A >_x B]$

The failure of the certificate $[A >_x B]$ indicates that A and B have changed their relative order due to the crossing in their x coordinates. So, only left(ζ_1) and right (ζ_2) neighbours are changed for both A and B. Before the event, A was the right neighbour to B and B was the left neighbour of A. But after the



(a) Safe region of each item after the violation of certificate $A >_x B$



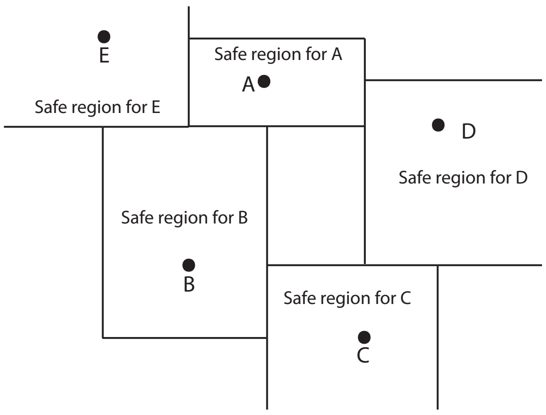
(b) Changes in graph according to the changes in safe regions

Figure 3.2: Failure of certificate $A >_x B$

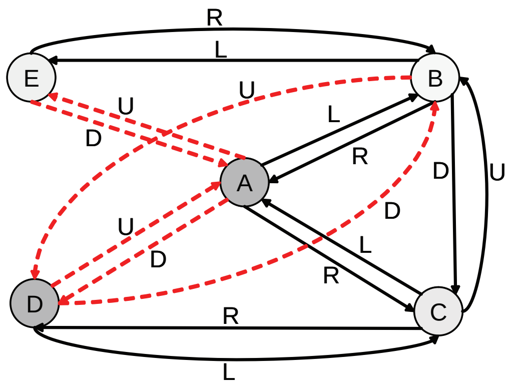
event they just form a reverse relationship. Moreover, the right neighbour of A becomes the right neighbour of B and the left neighbour of B becomes the left neighbour of A. So, to maintain the *view* we will have to update the graph accordingly. In addition to these, the left neighbour of B and right neighbour of A are also affected on failure of the certificate. The new right neighbour of the left neighbour of B becomes A and also the new left neighbour of the right neighbour of A becomes B (Figure 3.2b).

Event: Failure of Certificate $[A \ <_x \ B]$

If the site A moves to the right direction to cross the right boundary C, then the left neighbour of A and right neighbour of C are also affected on the failure of event. Besides these, new neighbouring relationship will be formed. Before events, C was the right neighbour of A. But after the event C will be the left neighbour of A. At the same time, A will be the right neighbour of C. Besides these, the left neighbour of A will be the left neighbour of C as well as the right neighbour of C will be the right neighbour of A. So, to maintain the *view* we will have update the graph accordingly.



(a) Safe region of each item after the violation of certificate $A >_y B$



(b) Changes in graph according to the changes in safe regions

Figure 3.3: Failure of certificate $A >_y B$

Event: Failure of Certificate $[A \ >_y \ B]$

After crossing the up neighbour D, A get new up neighbour (ζ_3) and down neighbour (ζ_4). Since, A goes above D, the new up neighbour of D will be the A and the same way, the new down neighbour of A will be the D. In addition to these, the up neighbour of A will be the neighbour of D which was on the up boundary of D before the event and the down neighbour of D will be the neighbour of A which was on the down boundary of A before the event (Figure 3.3b).

Event: Failure of Certificate $[A \ <_y \ B]$

If the site A crosses its down boundary point, A will get the site C as its new down neighbour which was B before the event. So, the new up neighbour of C will be the site A as well as the new down neighbour of D will be the site B. Therefore, B will get the site D as its up neighbour. On the other hand, since A goes below to the site B, the down neighbour of B will be the site A and the up neighbour of A will be the point B.

Pseudocode

Let us have a site A in π. A have 4 choices of direction to move forward and cross one of its 4 boundaries. It causes the violation of one of the certificates and immediately it needs to update the certificates associated with all the sites. If the boundary point of A, say B participates to make one of the certificate failures then the Algorithm 1 needs to be called.

For the sake of generalization, the adaptation process will start from least x-axis for one of the first two certificates failure and from least y-axis for one of the last two certificates failure.

Lemma 1. *For any instant of time t, if a certificate $c_i \in C$ fails then the Algorithm 1 takes O(1) time to update the configuration π_t to maintain the attribute* view.

Proof. From Algorithm 1, it is clearly shown that the failure of one certificate in any topological event needs exactly 4 certificates to be updated where each certificate update takes $O(1)$ time. Therefore, Algorithm 1 takes $O(1)$ time for updating the NG. □

3.4 Integrating Empty Circle Test

A variation of the definition of *view* in 3D is considered here. It is closely observed that if two point crosses each other from relatively far distance then a very small impact on natural view is made.

From this concept, we add another criterion in neighbourhood relationship. At the time of occurrences of any topological event by two points p_1 and p_2, the circle of diameter p_1p_2 contains no other site from P. Therefore, we have to check whether the circle is empty or not which can be done by the following steps:

Algorithm 1 Algorithm for updating the NG on certificate failure for a point A with B

```
BEGIN
if the certificate $[A <_x B]$ fails then
  $k \leftarrow 1$, $v_i \leftarrow A$, $v_j \leftarrow B$
else if the certificate $[A >_x B]$ fails then
  $k \leftarrow 2$, $v_i \leftarrow B$, $v_j \leftarrow A$
else if the certificate $[A <_y B]$ fails then
  $k \leftarrow 3$, $v_i \leftarrow A$, $v_j \leftarrow B$
else if the certificate $[A >_y B]$ fails then
  $k \leftarrow 4$, $v_i \leftarrow B$, $v_j \leftarrow A$
end if
$tempVertex \leftarrow \emptyset$ [Create a temporary vertex]
$tempVertex \leftarrow v_i.\zeta_k$
$v_i.\zeta_k \leftarrow v_j.\zeta_k$
$v_j.\zeta_k \leftarrow v_j.\zeta_{k+1}$
$v_j.\zeta_{k+1} \leftarrow v_i.\zeta_{k+1}$
$v_i.\zeta_{k+1} \leftarrow tempVertex$
return
END
```

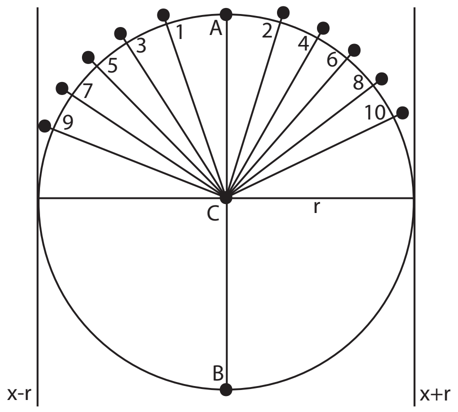


Figure 3.4: The worst case scenario on testing the empty circle

- Step 1: Consider the line (p_1, p_2) as the diameter of the circle. So radius $r = distance(p_1, p_2)/2$.
- Step 2: Find the ζ_1 and ζ_2 boundary point p_3 (or ζ_3 and ζ_4) of p_1 until the distance from p_3 to the center of the circle is not greater than r.

Algorithm 2 get_next_candidate(A, B, C, direction)

```
BEGIN
if $A.x = B.x$ then
   if $direction = 0$ then
      return $C.\zeta_1$
   else
      return $C.\zeta_2$
   end if
else
   if $direction = 0$ then
      return $C.\zeta_3$
   else
      return $C.\zeta_4$
   end if
end if
END
```

For details, we take a FIFO Queue, Q to hold all the candidate points to be checked. Lets during the moment of crossing of two points A and B, a circle with center C of diameter AB is calculated. We will insert the left neighbour of A and right neighbour of A to the Queue. Then we will check A whether it is in the circle or not. If not then we Enqueue the left neighbouring site of A into Q and check the right neighbour of A which was inserted in the previous cycle. If the right neighbour is not inside the circle then we will insert the right neighbour of that point and do the same process until the checking points is apart from center more than the radius of the circle in x-axis (y-axis in case of y-crossing). In worst case, there are infinitely many points p_i for which $|p_i.x - C.x| < r$ for the first two types of certificate failure and $|p_i.y - C.y| < r$ for the last two types of certificate failure as well as p_i is not inside the circle [see Figure 3.4]. The algorithm searches a point inside the circle from the middle of the circle dividing it into two parts. It then searches level wise to quickly decide whether it is empty or not. Algorithm 3 uses Algorithm 2 to search for the next candidate to insert into the Queue. The decision is taken on the left(up) and right(down) neighbourhood relationship of the point A or B (both have same x-axis or y-axis during certificate failure). Each time it toggles the choices between left(or up) neighbouring site and right (or down) neighbouring site.

Lemma 2. *At the time of the topological event by the two sites A and B, Algorithm 3 takes $O(k)$ time and $O(1)$ memory in worst case for testing the circle made from the diameter AB is empty.*

Proof. Let us have k number of points just above the circle of diameter AB similar to Figure 3.4. The Algorithm 3 must check all the k points to find any one inside the circle. So, the runtime complexity is just $O(k)$. However, in worst case both A and B are the extreme points placed in the boundary and the distribution of rest of the points are like Figure 3.4, i.e. $k = n - 2$. So, we can define k as an output sensitive parameter.

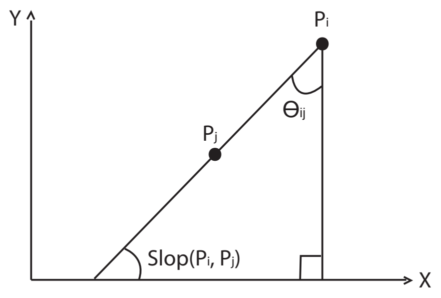
The Algorithm 3 uses a Queue Q to store the node to be checked. If we carefully observe, it is easy to see that only two nodes are stored at any instant of time. This is because, initially two nodes (left and right to A or B) are stored. In the loop, before insertion of any one of the left neighbour or right neighbour, a Dequeue operation takes place which remain the size of Q fixed. So, The Algorithm 3 needs a constant amount of memory and hence the memory complexity is $O(1)$. □

3.5 Proof Scheme

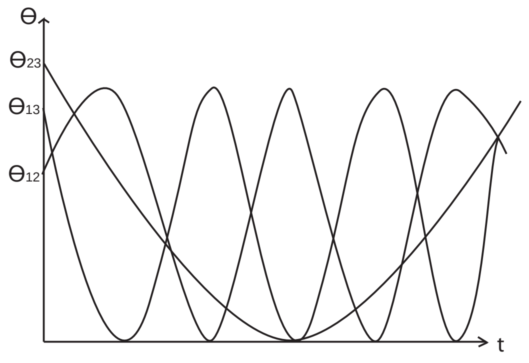
Let us consider what happens if the view points in 3D start moving with a polynomial function of time. For each step of time we get n points on the projection plane from the three dimensional space by applying the ray shooting

Algorithm 3 is_in_circle(NG, A, B)

```
BEGIN
$Queue\ Q \leftarrow \emptyset$
$dir \leftarrow 0$
$radius \leftarrow 0$ {Find the center C of the circle taking AB as diameter of the circle}
if A.x = B.x then
    $C.x \leftarrow (A.x + B.x)/2$, $C.y \leftarrow A.y$
    $radius \leftarrow |A.x - B.x|/2$
else
    $C.y \leftarrow (A.y + B.y)/2$, $C.x \leftarrow A.x$
    $radius \leftarrow |A.y - B.y|/2$
end if
$p \leftarrow get\_next\_candidate(A, B, A, 0)$
$q \leftarrow get\_next\_candidate(A, B, A, 1)$
EnQueue (Q,p)
EnQueue (Q,q)
while Q is not empty do
    $v \leftarrow DeQueue(Q)$
    if $distance(C, v) < radius$ then
        return 0
    end if
    if $|v.x - C.x| > radius$ AND $dir = 1$ then
        return 1
    end if
    EnQueue(get_next_candidate(A,B,A,dir))
    if dir = 0 then
        $dir \leftarrow 1$
    else
        $dir \leftarrow 0$
    end if
end while
return 1
END
```



(a) Calculates each Θ_{ij}



(b) The lower envelope from ${}^nC_2$ curves

Figure 3.5: Efficiency computation

from the view points. So, for a complete tour of the view point through a pre-defined trajectory, each point p_i in the projection plane will get a specific path of movement which can be defined as a continuous function of time. Let $(f_i)_{i=0}^{n}$ be the set of polynomial function of maximum degree s where each f_i belongs to each point $p_i(t) = [x_i(t), y_i(t)]$. If the discrete event is considered when two points cross each other either in x-coordinates or in y-coordinates, we can compute the angular distances between the line of two points (p_i, p_j) and the line $y_i(t) = x_i(t)$ at any instant of time t. For a pair of moving points in the projection plane, we define $\Theta_{ij} = \Delta(p_i(t), p_j(t)) mod\ 90$ where Δ represents the *Slop*.

Now, if we plot $\Theta_{ij}(t)$ for each i, j and time t, we get total ${}^nC_2$ number of curves (Figure 3.5b). We are interested to find all time t for which the curve touches the x-axis (i.e. $y = 0$). This can be found by solving all the equation of $\Theta_{ij}(t) = 0$. In fact, this is minimization diagram where maximum size of lower envelope is known by the Davenport-Schinzel Sequences.

Theorem 1. *The* Neighbourhood Graph *for maintaining the* view *in three dimensional Euclidean space has efficiency $O(\lambda_s(n^2))$ without circle test and $O(k\lambda_s(n^2))$ with circle test.*

Proof. In the worse case, for each Θ_{ij} for $1 \leq (i,j) \leq n$ we find the lower envelope[see Figure 3.5b] such that each I_i in the lower envelope contains exactly one intersection in $x-axis$ of the graph. So, total number of events will be the same as the maximum length of Davenport-Schinzel sequences, $\lambda_s(n)$. In our problem, total number of univariate functions F is ${}^nC_2$ which is $O(n^2)$. So, we have total $\lambda_s(n^2)$ number of internal events.

So, for our simple case the efficiency of the NG is $4\lambda_s(n^2)$ which is $O(\lambda_s(n^2))$. However, if we integrate the circle test we found the efficiency $O(k\lambda_s(n^2))$ because external events needs $O(k)$ time in worst case [See Lemma 2]. □

Theorem 2. *The proof scheme for the NG has locality* $O(1)$ *and responsiveness is* $O(1)$ *without circle test and* $O(k)$ *with circle test.*

Proof. Case I (without circle test): From Algorithm 1 we have seen that there are at most 4 certificates involved in any given item at any instant of time. So the *locality* of the KDS is $O(1)$. From Lemma 1 we know that the update procedure for failure of an certificate takes $O(1)$ runtime. So, the *responsiveness* of the NG is $O(1)$.

Case II (with circle test): If we consider empty circle as one certificate than total 5 certificates are needed to certify the attributes. So still the *locality* of the NG is $O(1)$. From Lemma 2, it is proved that in worst case, the Algorithm 3 takes $O(k)$ times to decide. So, the *responsiveness* of the NG is $O(k)$. □

Chapter 4

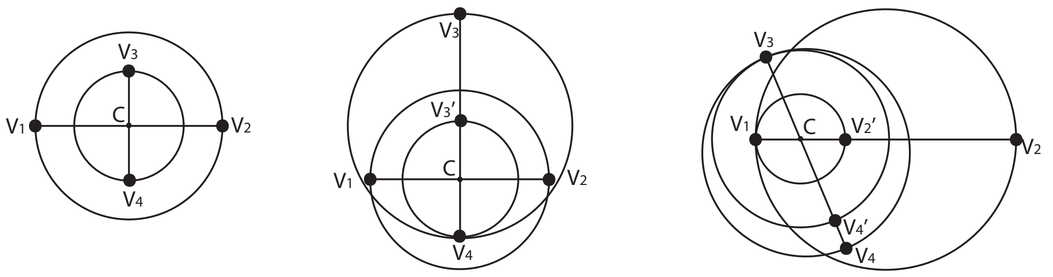
Empty Circle Revisited

In the previous Chapter, it is shown that the response time for the circle test is $O(k)$. There exist a configuration where all the points in the point set are candidate to test whether any of them are inside the circle. Thus, when $k = n-2$, the response time become $O(n)$. So, a comprehensive study has been done on a graph constructed from the empty circles and shown that the total number of empty circles in a set of point is linear. The empty circle graph is well known as Gabriel graph which is introduced by Gabriel and Sokal [12]. In this Chapter, it is proved that the Gabriel graph is connected as well as planar. So, naturally, one of the upper bound of the number of empty circles in the Gabriel graph is $O(3n-6)$. In the last section, we show another bound of the number of edge in Gabriel graph which is $O(3n-8)$.

4.1 Some Important Properties

Definition 8. EMPTY CIRCLE*: Let $P = \{p_1, p_2, p_3, \ldots, p_n\}$ be a set of points in two dimensional plane $\mathbb{R}^2$. For any pair of points (p_i, p_j) $\in P$ where $i \neq j$, if it is possible to construct a circle considering p_ip_j as diameter such that no other point from $P - \{p_i, p_j\}$ is either inside or on the circle then p_ip_j can be called as empty circle.*

Definition 9. GABRIEL GRAPH*: Let $GG = (V, E)$ be a graph where V and E*



(a) Case I (b) Case II (c) Case III

Figure 4.1: Gabriel graph is planar

represents the set of vertex and edge respectively. GG is said to be an Gabriel graph if each point $p \in P$ is considered as the vertices and each empty circle is considered as its edge in GG. More specifically, for a Gabriel graph GG, there is an edge between v_i and v_j if and only if the circle v_iv_j is empty.

Matula et al. [18] in their paper on geographic variation research showed if there two edges AB and CD in the Gabriel graph intersect at an interior point then one of the internal angle in the quadrilateral $ACBD$ must be greater than or equal to $90°$. So, Gabriel graph is actually a planar graph. An alternative proof of planarity of Gabriel graph is given below:

Lemma 3. *Gabriel graph is planar*

Proof. To prove the planarity of Gabriel graph by contradiction, we assume that the Gabriel graph is not planar. So, it is possible to find two edge v_1v_2 and v_3v_4 in $E(GG)$ that intersect each other at any point C. Therefore, it is sufficient to give counter example for all possible cases which contradicts with the assumption.

- Case I: Let us consider two edges v_1v_2 and v_3v_4 intersect at point C in such a way that each of the two edges are the bisector to another. If the length of the edge v_1v_2 and v_3v_4 are equal [see Figure 4.1a] then four vertices v_1, v_2, v_3 and v_4 are co-circular on a circle with center C and radius $r = Cv_1 = Cv_2 = Cv_3 = Cv_4$. Therefore, no such edge v_1v_2 and v_3v_4

should be formed. In another case where the length of v_1v_2 and v_3v_4 are not equal, because of the center of both circles v_1v_2 and v_3v_4 are same, the circle with smaller radius must inside the circle with greater radius. Both of the above conditions forbid the greater circle to be formed and no such edge in the Gabriel graph can be found. Hence, our assumption is contradictory.

- Case II: Now in the second case, let v_1v_2 and v_3v_4 intersect in C such a way that C is the middle point for v_1v_2 edge only. In other words we can say that the edge v_3v_4 becomes the bisector of the edge v_3v_4 but the vice versa is not necessarily true [see Figure 4.1b]. Now, we obtain a temporary vertex $v_3{}'$ such that the length of Cv_3' is equal to the length of Cv_4. Therefore, C becomes as the middle point for both the segment v_1v_2 and $v_3'v_4$. So, according to the Case I, v_4 must inside the circle v_1v_2 because the length of $v_3'v_4$ is less than the length of v_1v_2. Hence, v_1v_2 edge is not possible.

- Case III: Finally, consider the situation where two edges intersect in C such a way that no edge is the bisector for another edge [see Figure 4.1c]. In this case, we find two nearest vertex of C, one from v_1v_2 and another from v_3v_4. According to the Figure 4.1c v_1 and v_3 are two nearest points of C. Now we consider two temporary points v_2' and v_4' such that v_1v_2' and v_3v_4' becomes the bisector for each other. Therefore, C is the center of both the circle of v_1v_2' and v_3v_4'. Thus this is exactly the Case I. If the length of Cv_1 is greater than the length of Cv_3 then vertex v_3 must inside the circle v_1v_2'. So v_3 is inside v_1v_2 as well and v_1v_2 must not form an edge in Gabriel graph. Hence the assumption we made is contradictory. This

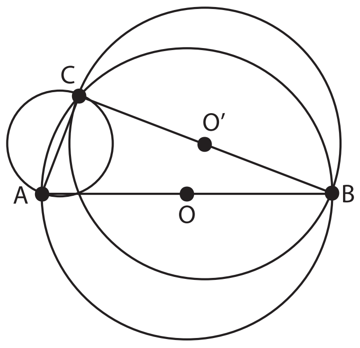


Figure 4.2: Removal of empty circle AB causes the creation of AC and BC

also holds when the length of the segment Cv_1 is less than or equal to the length of Cv_3.

So, according to the definition of Gabriel graph and its properties, we do not have such two edges which can intersect each other. Therefore, Gabriel graph is planar.

□

Lemma 4. *Let us consider a Gabriel graph with only two vertices A and B and one edge AB. Now, if a third point C is inserted inside or on the circle AB then the edge AB will be removed and two new edges AC and BC must be formed into the corresponding Gabriel graph.*

Proof. If the new point C is inserted into the circle AB, then the edge AB will be removed from the Gabriel graph. We have to prove that both the circle AC and BC must be empty. More specifically, the circle AC does not contain the point B and circle BC does not contain the point A. Consider that the newly inserted point C is very closer to the point A on the circle AB and O' is the center of the circle BC [see Figure 4.2]. So, our goal is to find whether there exits any position C on the circle AB for which the circle BC contains the point A. If we can prove that for any C the inequality conditions of $O'A > O'C$ never violets, then A will never be on or inside the circle BC.

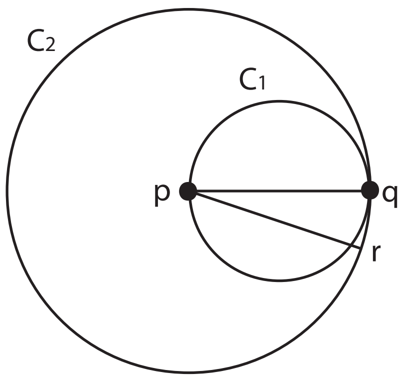


Figure 4.3: Nearest neighbour form empty circle

We know that the diameter is the biggest chord of a circle. So, the length of chord AB is greater than the length of chord BC and $O'B < OB$. We can also say that $OA > O'C$. Since, $O'A > OA$, The inequality $O'A > O'C$ holds until O' does not overlap with O which happen only when C overlap with B. Hence, circle BC never contain A. Similarly it can be proved that the circle AC is also empty. □

Lemma 5. *Let p and q are two points from a set of n points P. If q is one of the nearest neighbour of p then pq is an Empty Circle and there exists an edge pq in the Gabriel graph.*

Proof. To prove the lemma, we can construct a circle C_1 considering the point p as center and pq as radius [see Figure 4.3]. Since, q is one of the nearest points of p, no points except p and q can be found inside the C_1. if there exists multiple nearest point to p then every point is on the periphery of C_1. Now, lets construct another circle C_2 considering pq as diameter. We can see that C_2 must inside of C_1 and so C_2 is empty as well. Now we can claim that if the circle C_2 intersect C_1 at a single point q then C_2 is certainly an empty circle. This is because if C_2 intersect C_1 at another point s from P other than q, then s may also be another nearest neighbour of p. Therefore, s exists both on circle C_1 and C_2 which makes C_2 not empty.

We can prove the above claim by the method of contradiction. We assume that C_2 intersect C_1 at another point s which is very closer to q. Since pq is the diameter and ps is a chord in C_2, $ps < pq$. However, both ps and pq are the radius of C_1 so $ps = pq$. This is contradiction. So, q is the only point at which C_2 and C_1 intersect. $\square$

Although Gabriel et al. [12] already proved that Gabriel graph is a connected graph, we provide alternative and simple arguments to establish Gabriel graph as a connected graph.

Lemma 6. *Gabriel graph is connected.*

Proof. Let us have a disjoint set of n components $A = \{A_1, A_2, A_3, \ldots, A_n\}$ where each $A_i \subseteq GG$ for $1 \leq i \leq n$. Initially each component consists of one vertex from the set of vertex $V(GG)$. For each steps, we try to connect two neighbouring component maintaining the constraint of Gabriel graph until all the components are connected. We define an edge (u, v) as *safe edge* between two components A_i and A_j such that $u \in A_i$, $v \in A_j$ and uv circle (circle with the diameter uv) is empty. Now, it can be said that if there is at least one safe edge between two disjoint components then they must be connected in Empty Circle Graph.

We define the nearest vertex pair between two components A_i and A_j as $Nv(A_i, A_j)$ and the nearest component of a particular component A_i as $Nc(A_i)$. Lets $minVdis(A_i, A_j)$ is the distance between two nearest point between two components A_i and A_j and $minCdist(A_i)$ is the distance between A_i and its nearest component. So, we can define $Nv(A_i, A_j)$ and $Nc(A_i)$ by the following ways:

$$minVdist(A_i, A_j) = minimal\{\forall_{u \in A_i, v \in A_j}(||u - v||)\}$$
$$Nv(A_i, A_j) = \{(u, v) : \exists_{u \in A_i, v \in A_j}(||u - v|| = minVdist(A_i, A_j))\}$$

$minCdist(A_i) = minimal\{\forall_{1 \leq j \leq n, j \neq i}(min(A_i, A_j))\}$

$Nc(A_i) = \{A_j \in A : \exists_{1 \leq j \leq n, j \neq i}(min(A_i, A_j) = minCdist(A_i))\}$

We claim that if the nearest vertex pair between a component A_i and its nearest component A_j is (u, v) then u and v are also the nearest neighbour to each other. Because, if there were another neighbouring vertex s from another component A_k which is more closer from u then A_j would not be the nearest component of A_i. So, according to the Lemma 5, the edge (u, v) must be safe.

Now, we start our process by selecting one of the n components $A_1 \in A$ and then find its nearest component, A_2. We can have $n - 1$ components by connecting A_1 and A_2 through their safe edge. we continue this process by connecting the nearest component of the newly formed connected component A_1A_2 until no other component left other than the newly connected component.

□

4.2 Lower Bound and Upper Bound

We know, a connected planar graph with n vertices can have minimum $n - 1$ number of edges and maximum $3n - 6$ number of edges. So, this is also a bound of the number of edges in Gabriel graph. However, Matula et al. [18] give another tighter upper bound which is $3n - 8$. In rest of the part of this chapter, we proved that lower and upper bound in slightly differently ways than the way Matula et al. [18] did.

Lemma 7. *Lower bound on the number of edges of an Gabriel graph on the plane $\mathbb{R}^2$ is $n - 1$.*

Proof. In Lemma 6, we established that Gabriel graph is a connected graph and thus no point is disconnected (isolated). Let P be a set of n points in the plane $\mathbb{R}^2$. To prove the theorem, we construct an Gabriel graph GG with minimal

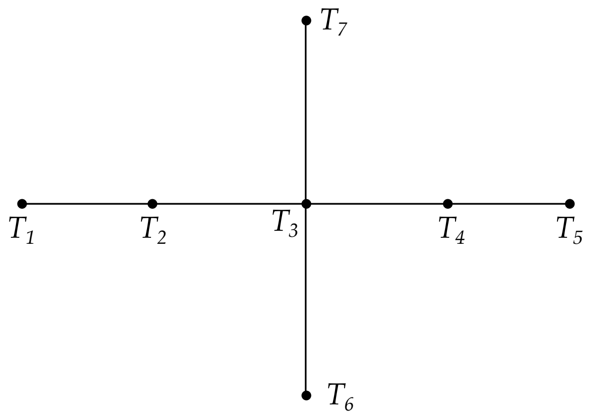


Figure 4.4: One of the lower bound configuration

number of edges by adding one point at a time from the set P. The first point in GG does not produce any edge but all other subsequent points can be placed in GG with minimal one edge for each point (since no point is disconnected). Thus n points of P can be placed in GG with lowest $n-1$ edges (empty circles) as shown in the Figure 4.4. □

Definition 10. *The vertices and edges incident with the external or unbounded face of a planar graph are respectively referred as external vertices and external edges of the graph. The number of external vertices and edges of a planar graph G are respectively denoted as $\Omega_v(G)$ and $\Omega_e(G)$.*

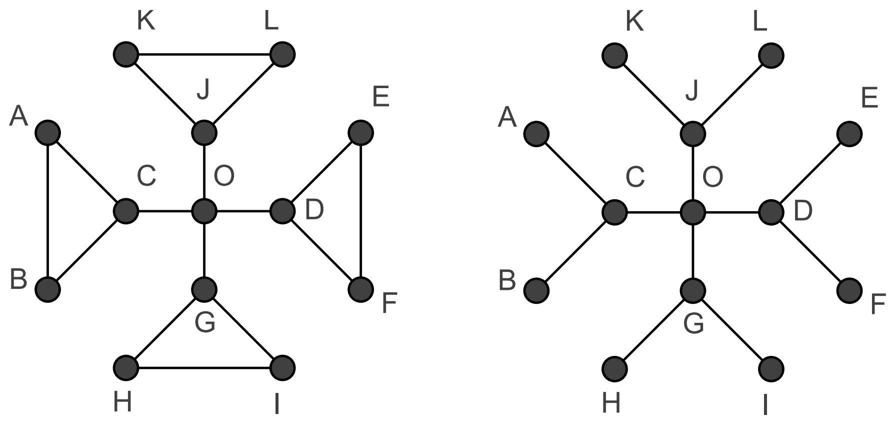


Figure 4.5: The graph on the left contains cycles and the graph on the right does not contain cycle.

Theorem 3 ([31]). *Every connected simple planar graph G with $v \geq 2$ vertices has at most $3(v-1) - \Omega_v(G)$ edges.*

Proof. Suppose that G is a connected simple planar graph, with v vertices, e edges, and f faces. Then Euler's formula states that,

$$v - e + f = 2 \tag{4.2.1}$$

In a connected simple planar graph any face except the unbounded face is bounded by at least three edges and every edge is incident with two faces. Thus we can write $2e \geq 3f$. If we consider the half edges incident with the bounded faces of G, then the previous inequality can be given in the following form,

$$2e - \Omega_e(G) \geq 3(f-1) \tag{4.2.2}$$

Here the edges $\Omega_e(G)$ incident with the unbounded face are also incident with some other bounded faces and included in the total edges e of the graph G. Therefore, using Euler's formula we get,

$$e \leq 3(v-1) - \Omega_e(G) \tag{4.2.3}$$

There are two cases to consider depending on whether the graph G contains cycle or not. If the graph G contains at least one cycle, then the number of external edges $\Omega_e(G) \geq \Omega_v(G)$ for the graph G. Since, each cycle in the graph G provides an additional edge with the minimum number of edges $v-1$ of a connected graph as illustrated in Figure 4.5. The vertices and edges inside the cycles can be discarded for this purpose since these vertices and edges are not external vertices and external edges. Thus the graph G has at least three vertices and we can write $e \leq 3(v-1) - \Omega_v(G)$. On the other hand, if the graph G does not contain any cycle, then the graph G is a tree. For any tree G, we know that the total number of edges $e = v - 1$ and all the vertices of G are incident with the unbounded face. Therefore, $v - 1 \leq 3(v-1) - v$ for all

$v \geq 2$ and we can write $e \leq 3(v-1) - \Omega_v(G)$ for the tree G. This completes the proof. □

Corollary 2 ([31]). *Every connected simple planar graph G with $v \geq 2$ vertices has at most $2v - 1 - \Omega_v(G)$ faces.*

Proof. The upper bound on faces $f \leq 2v - 1 - \Omega_v(G)$ of the graph G follows from Theorem 3 using Euler's equation among vertices, edges and faces of the planar graph G. □

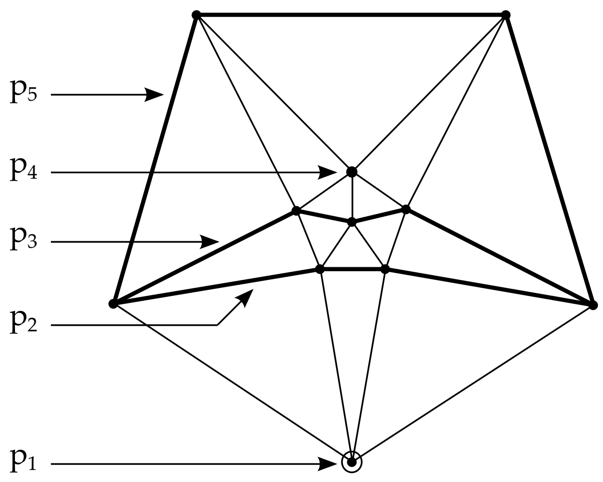


Figure 4.6: One of the upper bound configuration

Since the Gabriel graph, GG is a connected simple planar graph, it also satisfies the inequality $3(v-1) - \Omega_v(GG)$ following from Theorem 3. It is clear from the inequality that the minimizing the value of $\Omega_v(GG)$ obtains the maximization of the term $3(v-1) - \Omega_v(GG)$. Thus a nested arrangement of points with minimum exterior vertices provides the upper bound value 4.6. Since a Gabriel graph GG can have no vertex interior to any triangle or quadrilateral, a Gabriel graph GG with $v \geq 5$ vertices has at most E^* edges.

$$E^* \leq 3(V-1) - 5 \tag{4.2.4}$$

$$\leq 3V - 8 \tag{4.2.5}$$

Chapter 5

Spatial Subdivision of Gabriel Graph

The objective of this Chapter is to propose a technique of subdivision of the domain of Gabriel graph spatially. The continuous movement of a point is modeled into the computer by some repeated discrete deletions and insertions. Handling the repeated deletions and insertions to maintain the attribute *view*, a special type of data structure is required. As a solution, the whole region of Gabriel graph can be divided into a set of cells named *sparse cells*. The advantages of this cellular arrangement is: it is possible to avoid some computation for a moving point while staying in a specific cell. Here, for each points p_i, an *effective region* $R(p_i)$ has been defined in such a way that if a new point p_k is inserted into $R(p_i)$ then it will be connected to p_i and will form an edge p_ip_k in the Gabriel graph. Another type of cell named *effective cell* is derived from the *sparse cells* and *effective regions* which is actually the intersection of a set of *effective regions*. In this Chapter, construction of the *effective regions* are described and the necessary algorithms for updating the *effective regions* are provided for each insertion or deletion of a point in the graph. The following theorem is established to get the $O(M\log(M)$ run time solution for testing the emptiness of a circle for maintaining the *view*.

Theorem 4. *The query for the existence of a Gabriel edge can be answered in* $O(\log(M))$ *time and including the update cost it takes total* $O(M\log(M))$ *times.*

5.1 Sparse Cell

The following are the equivalent definitions for Gabriel graph [12]:

1. AB is an edge of the Gabriel graph GG *iff* the $\angle ACB$ is acute for every $C \in V(GG), C \neq A, C \neq B$.

2. The Vertices $\{A, B\} \in V(GG), A \neq B$ are least squares adjacent forming an edge AB *iff*

$$d_{AB}^2 < d_{AC}^2 + d_{BC}^2 \; for\ all\ C \in V, C \neq A, C \neq B.$$

   The vertices and pairs of least square adjacent vertices (edges) determine the least square adjacency graph $G(V)$. if V is a set points in the plane and d_{AB} denotes the Euclidean distance, the $G(V)$ is a Gabriel graph $GG(V)$.

Therefore, it is clear that 90° is the angle limit which determines the other vertices eligibility to construct edges with the adjacent vertices. Thus the orthogonal lines on edges through vertices divide the plane into regions with similar edge construction properties of Gabriel graph. Let p_i, p_j be two points in $\mathbb{R}^2$ and an orthogonal line $L(\widehat{p_i}, p_j)$ to the line segment p_ip_j passing through p_i divide the plane into two half planes. The right open half plane $ROH(L(\widehat{p_i}, p_j))$ contains the part of the plane enclosed p_j and the left closed half plane $LCH(L(\widehat{p_i}, p_j))$ contains the rest of the plane. Similarly, the left open half plane $LOH(L(p_i, \widehat{p_j}))$ containing p_i and the right closed half plane $RCH(L(p_i, \widehat{p_j}))$, can be found by considering $L(p_i, \widehat{p_j})$. The regions constructed by partitioning the plane $\mathbb{R}^2$ based on a line segment p_ip_j can be defined as

$$\begin{aligned}
R(\widehat{p_i}, p_j) &= LCH(L(\widehat{p_i}, p_j)) \\
&= LOH(L(p_i, \widehat{p_j})) - ROH(L(\widehat{p_i}, p_j)) \\
R(p_i, \widehat{p_j}) &= RCH(L(p_i, \widehat{p_j})) \\
&= ROH(L(\widehat{p_i}, p_j)) - LOH(L(p_i, \widehat{p_j})) \\
R(\widehat{p_i}, \widehat{p_j}) &= ROH(L(\widehat{p_i}, p_j)) \cap LOH(L(p_i, \widehat{p_j}))
\end{aligned}$$

Thus union of the set $R(p_i, p_j)$ of these three regions produce the $\mathbb{R}^2$ plane and the set $R(p_i, p_j)$ can be given as

$$R(p_i, p_j) = \{R(\widehat{p_i}, p_j), R(\widehat{p_i}, \widehat{p_j}), R(p_i, \widehat{p_j})\}$$

Two other regions $\overline{R(p_i, \widehat{p_j})}$ and $\overline{R(\widehat{p_i}, p_j)}$ can also be defined using the combination of the predefined regions as follows

$$\overline{R(p_i, \widehat{p_j})} = \mathbb{R}^2 - R(p_i, \widehat{p_j}) = R(\widehat{p_i}, p_j) \cup R(\widehat{p_i}, \widehat{p_j})$$
$$\overline{R(\widehat{p_i}, p_j)} = \mathbb{R}^2 - R(\widehat{p_i}, p_j) = R(p_i, \widehat{p_j}) \cup R(\widehat{p_i}, \widehat{p_j})$$

A *sparse cell* is the non empty effective region $h \subset \mathbb{R}^2$ constructed from the common intersection of the each unique region r such that $r \in R(p_i, p_j)$ and $h \subseteq r$ for every $p_ip_j \in E(DG)$. Formally, *sparse cell* sc can be defined as

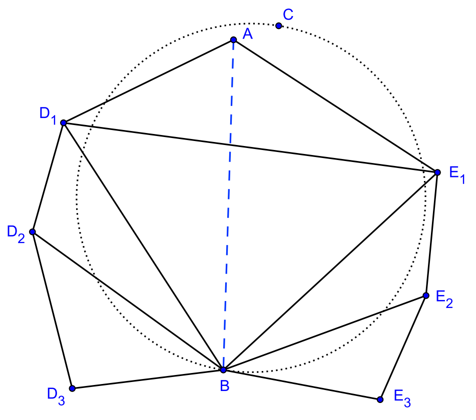
$$\begin{aligned} sc = \bigcap \{r : \forall p_ip_j (p_ip_j \in E(DG) \\ \implies \exists! r (r \in R(p_i, p_j) \wedge h \subseteq r))\} \end{aligned}$$

where sc cannot be empty. For an example, the *sparse cell* R_1 in Figure 5.3 can be given as

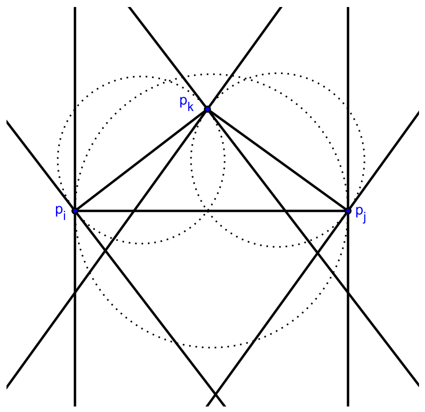
$$\begin{aligned} R_1 = \bigcap \{R(\widehat{p_1}, \widehat{p_2}), R(\widehat{p_1}, \widehat{p_3}), \\ R(\widehat{p_2}, p_3), R(p_1, \widehat{p_4}), R(\widehat{p_2}, \widehat{p_4})\} \end{aligned}$$

These bounded and unbounded convex *sparse cells* joined together to form the cell complex of an arrangement which can be constructed by the boundary lines of the regions by Delaunay edges.

It is claimed that an edge p_ip_j is not an Gabriel edge if and only if there exists a point $p_k \in V(GG)$ for which $p_j \in R(p_k, \widehat{p_i})$. This is actually the basis of our proposed subdivision of Gabriel graph. We can express the previous statement in slightly different way: if we insert a point p_j in any location in the Gabriel graph and if p_i has any such Gabriel neighbour p_k where $p_j \in R(p_k, \widehat{p_i})$, then p_ip_j must not be an edge in GG. So, our goal is to find a specific region where there is no such p_k for p_i. With the help of Lemma 8 and Lemma 9 we give the Theorem 5 to provide such specific region or cells in the space.



(a) Contradictory to Lemma 8



(b) Proof of Lemma 9

Figure 5.1: p_j must in $R(p_k, \widehat{p_i})$

Lemma 8. *For any two vertices* $\{p_i, p_j\} \in V(GG)$*, if the edge* $p_ip_j \notin E(GG)$ *there exist a vertex* $p_k \in V(GG)$ *inside the circle* p_ip_j *such that* $p_ip_k \in E(DG)$*.*

Proof. According to the definition of Gabriel graph it is obvious that if $p_ip_j \notin E(GG)$ then there exist a set of points $S \subseteq V(GG) - \{p_i, p_j\}$ inside the circle p_ip_j where $S \neq \emptyset$. So it is sufficient to proved that there exists a point $p_k \in S$ for which $p_ip_k \in E(DG)$.

we start with a contradictory scenario similar to the Figure 5.1a where $p_i = B$, $p_j = C$. We consider that there is exactly one point A inside the circle BC and the edge AB is not in Delaunay graph. It is clear that B is inside the circle AD_iE_j where each D_i and E_j are outside the circle BC and every pair of D_i and E_j are opposite to each other relative to AB. So, no edge can be found to flip AB unless otherwise specified. Therefore, our assumption is wrong and AB must be an edge in Delaunay graph. Thus $p_k = A$ and the Lemma holds.

Now, considering AB as diagonal, we get a quadrilateral AD_1BE_1 where two vertices D_1 and E_1 both are outside the circle BC as well as opposite to diagonal AB in the Delaunay graph. if we relax our scenario by putting D_1 inside the circle BC then AB can be flipped into D_1E_1 and therefore we get another quadrilateral $D_1D_2BE_1$. Again we can say that B is inside the circle $D_1D_iE_j$ where for any D_i and E_j outside the circle BC where every pair of D_i

and E_j are opposite to each other relative to D_1B. So, the edge $E_1D_2 \notin E(DG)$ therefore, BD_1 must be an Delaunay edge. if we continuously relax the scenario by putting every point D_i inside the circle BC to flip BD_{i-1} recursively, we get $BD_i \in E(DG)$. So, lemma holds if we consider $p_k = D_i$.

It should also be noted that if E_j is inside the circle BC then similarly it is very easy to prove that for $p_k = E_j$ the lemma also holds. In this case, all D_i can be either outside or inside the circle BC without violating the lemma. So, D_i or E_j or both can be inside the circle BC to flip either BD_{i-1} or BE_{j-1} but in every case BD_i or BE_j or both exists in the Delaunay graph. In other words, since, both E_i and D_i are not outside the circle BC at the same time, we can ensure at least one point p_k is inside the circle BC where $p_ip_k \in E(DG)$ □

Lemma 9. *For any two vertices $p_i, p_j \in V(GG)$, the edge $p_ip_j \notin E(GG)$ iff there exist a vertex $p_k \in V(GG)$ and $p_ip_k \in E(DG)$ such that $p_j \in R(p_k, \widehat{p_i})$.*

Proof. By Lemma 8, if $p_ip_j \notin E(GG)$ then there exists at least one p_k for which $p_ip_k \in E(DG)$. Since, p_k is inside the circle p_ip_j, $\angle p_ip_kp_j$ is not acute. So, the $\angle p_ip_kp_j - \angle p_ip_kp_l > 0$ for any point p_l on the perpendicular line to p_ip_k passes through p_k. So, this perpendicular line must intersect with the segment p_ip_j[Figure 5.1b]. So, $p_j \in R(p_k, \widehat{p_i})$. □

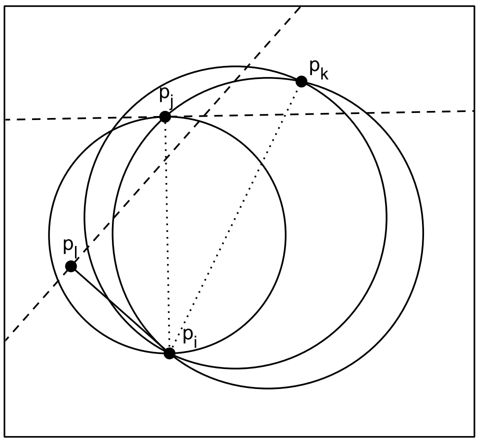


Figure 5.2: Gabriel edges are not enough

Howe [19] showed in this Ph.D. thesis that the Gabriel graph GG is actually the sub-graph of the Delaunay graph and it is possible to construct GG from DG.

Lemma 10 ([19]). *The Gabriel graph on a vertex set V is a sub-graph of the Delaunay triangulation for V. Furthermore, the edge AB of the Delaunay triangulation is an edge of the Gabriel graph iff the straight line joining A to B intersects the boundary line segment common to the Theissen polygons for A and B at a point other than the endpoints of that boundary line segment.*

Proposition 1. *There may exist vertices $p_i, p_j, p_k, p_l \in V(GG)$ such that $p_k \in R(p_i, \widehat{p_j})$ and $p_k \notin R(p_i, \widehat{p_l})$ and $p_ip_j \notin E(GG)$.*

Proof. Let us consider the vertices of Gabriel graph as shown in Figure 5.2. If p_l is inside the circle p_ip_j and outside the circle p_ip_k, then according to the fundamental properties of Gabriel graph $\angle p_ip_lp_k$ is acute. Thus $p_k \in R(\widehat{p_i}, \widehat{p_l})$ or $p_k \in R(\widehat{p_i}, p_l)$ that is $p_k \notin R(p_i, \widehat{p_l})$. The edge $p_ip_j \notin E(GG)$ since p_l is inside the circle p_ip_j. □

Thus, for any two vertices $p_i, p_j \in V(GG)$, the edge $p_ip_j \notin E(GG)$ *iff* there exist a vertex $p_k \in V(GG)$ and $p_ip_k \in E(GG)$ such that $p_j \in R(p_k, \widehat{p_i})$, is not true for all cases which is a modified form of Lemma 9.

Theorem 5. *if a point is inserted anywhere in a sparse cell then the set of Gabriel adjacent vertices of that point remain unique.*

Proof. Let a *sparse cell* sc of *effective region* $h \subset \mathbb{R}^2$ be the common non empty intersections of elements of R which can be written as

$$\begin{aligned} R = \{r : \forall p_ip_j(p_ip_j \in E(DG) \\ \implies \exists! r(r \in R(p_i, p_j) \wedge h \subseteq r))\} \end{aligned}$$

The order of p_ip_j is consistent throughout the proof. Let $R_r \subseteq R$ be defined as follows

$$R_r = R \cap \{R(p_i, \widehat{p_j}) : p_ip_j \in E(DG)\}$$

Thus a point p_k inserted anywhere in the *sparse cell* sc has same R_r. Since p_k is adjacent to all $p_i \in V(DG)$ except when $p_ip_j \in E(DG)$ for any vertex p_j and $p_k \in R(p_i, \widehat{p_j})$ according to the Lemma 8, the set of adjacent vertices V_k and non adjacent vertices V_r of p_k can be written as

$$V_r = \{p_i : \exists p_j (R(p_i, \widehat{p_j}) \in R_r)\}$$

$$V_k = V(DG) - V_r$$

Thus the set of adjacent vertices V_k remain same throughout the *sparse cell* since V_r only depends on R_r which is a subset of R. □

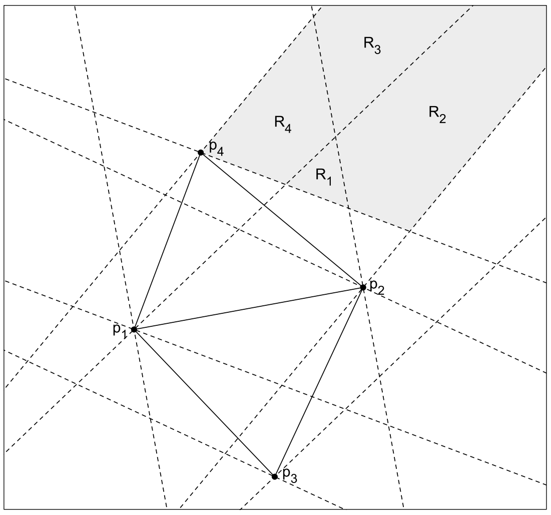


Figure 5.3: Share same set of adjacent

Proposition 2. *A unique set of adjacent vertices can be found if a point is inserted in different sparse cells.*

Proof. Let us consider a Gabriel graph GG of vertex set $P = \{p_1, p_2, p_3, p_4\}$ as shown in Figure 5.3. A point inserted into R_1 *sparse cell* has adjacent set $\{p_2, p_4\}$. Similarly, a point inserted into either R_2, R_3 or R_4 cell has the same adjacent set $\{p_2, p_4\}$. Thus different *sparse cells* can have a unique set of adjacent vertices.

□

5.2 Effective Region

The *effective region* of a vertex $p_i \in V(DG)$ is the common intersection of all the regions $\overline{R(p_i, \widehat{p_j})}$ such that $p_i p_j \in E(DG)$ from p_i. The *effective region* $R(p_i)$ of vertex $p_i \in V(DG)$ can be formalized as

$$R(p_i) = \bigcap \left\{ \overline{R(p_i, \widehat{p_j})} : \forall p_j (p_i p_j \in E(DG)) \right\}$$

The complement of the *effective region* $\overline{R(p_i)}$ of a vertex $p_i \in V(DG)$ can be defined as

$$\overline{R(p_i)} = \mathbb{R}^2 - R(p_i)$$

If the union of the set $R_\pi(p_i)$ induces the plane $\mathbb{R}^2$, then the set $R_\pi(p_i)$ can be defined as

$$R_\pi(p_i) = \{R(p_i), \overline{R(p_i)}\}$$

In the trivial case, if a point p_k is inserted into either in $R(\widehat{p_i}, p_j)$ or in $R(\widehat{p_i}, \widehat{p_j})$ then $p_k p_i$ can be an edge in Gabriel graph. However, if p_k is inserted in $R(p_i, \widehat{p_j})$ then $p_i p_k \notin E(GG)$. So, each of the $M(p_i)$ number of Delaunay neighbours p_r of p_i partition the plane into two half planes and the intersection of the half planes inscribed the point p_i comprise a convex polygon. The convex polygon can be called as the *effective region* of p_i because if any point p_k is inserted into the *effective region* of p_i then $p_i p_k$ must be connected.

A region $\overline{R(p_i, \widehat{p_j})}$ for some p_i and p_j is essentially a half plane bounded by the line $L(p_i, \widehat{p_j})$. If two half planes are not parallel, then their common intersection induces a vertex which is the intersection point of the associate lines of the half planes. This intersection point is denoted as intersection vertex of the half planes or regions. The edges from a vertex and so the orthogonal lines on the edges are not parallel in DG according to the general assumption.

Lemma 11. *The vertices of an* effective region $R(p_i)$ *are the intersection vertices of all the regions* $\overline{R(p_i, \widehat{p_j})}$ *and* $\overline{R(p_i, \widehat{p_k})}$ *such that the* $\angle p_j p_i p_k$ *is minimum for all distinct* $\{p_i p_j, p_i p_k\} \in E(DG)$.

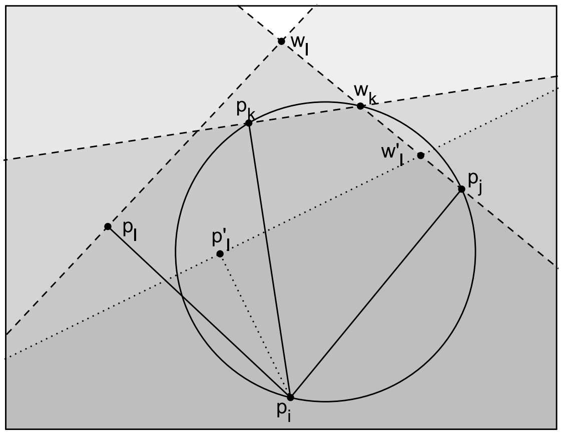


Figure 5.4: Region vertices

Proof. Assume for the purpose of contradiction that one of the vertices of the *effective region* $R(p_i)$ is the distinct intersection vertex w_l of the regions $\overline{R(p_i, \widehat{p_j})}$ and $\overline{R(p_i, \widehat{p_l})}$ such that the $\angle p_j p_i p_l$ is not minimum and $\{p_i p_j, p_i p_l\} \in E(DG)$ as shown in Figure 5.4. Thus, there exists $p_i p_k \in E(DG)$ for which $\angle p_j p_i p_k$ is minimum and w_k is the intersection vertex of the regions $\overline{R(p_i, \widehat{p_j})}$ and $\overline{R(p_i, \widehat{p_k})}$. Since the $\angle p_i p_k w_k$ is a right angle, $p_i w_k$ is the diameter of the circle $p_i p_j p_k$ and according to the alternative definition of Gabriel graph [12] the $\angle p_i p_l w_k$ formed with a point p_l outside the circle $p_i p_j p_k$ is acute. There are three regions of point p_l point need to be considered. First, if p_l is outside the circle $p_i p_j p_k$, then the length $w_k w_l$ can be determined using the sine rule $K \sin(\pi/2 - \angle p_i p_l w_k)$ for some constant K. Thus w_l is not a vertex of region $\overline{R(p_i, \widehat{p_j})} \cap \overline{R(p_i, \widehat{p_k})} \cap \overline{R(p_i, \widehat{p_l})}$ and clearly w_l is not a vertex of $R(p_i)$ contradicting the assumption of w_l. Second, If p_l is on the circle $p_i p_j p_k$, then the vertex of intersection $\overline{R(p_i, \widehat{p_j})}$ and $\overline{R(p_i, \widehat{p_l})}$ is w_k which is the vertex of intersection $\overline{R(p_i, \widehat{p_j})}$ and $\overline{R(p_i, \widehat{p_k})}$. This contradicts the distinction of w_l vertex. Third, if p_l is inside the circle $p_i p_j p_k$ as p'_l in Figure 5.4 then, according to the properties of Delaunay triangulation $p_i p_k \notin E(DG)$ which contradict the assumption that $p_i p_k \in E(DG)$. Similarly, we can prove the lemma for any point p_l located on the other side of line $p_i p_j$.

□

Lemma 12. *An edge $p_i p_j \in E(GG)$ forms* iff *a point p_j is inserted into the* effective region $R(p_i)$ *of* $p_i \in V(GG)$.

Proof. If a point $p_k \in R(p_i)$, then p_k must be in the common intersection of all the region $\overline{R(p_i, \widehat{p_j})}$ for all $p_ip_j \in E(DG)$ from p_i according to the definition of *effective region* $R(p_i)$. Thus there is no $p_ip_j \in E(DG)$ from p_i for which $p_k \in R(p_i, \widehat{p_j})$. From Lemma 9, we know $p_ip_k \notin E(GG)$ *iff* there exist a vertex $p_j \in V(DG)$ with $p_ip_j \in E(DG)$ for which $p_k \in R(p_i, \widehat{p_j})$ but there is no such $p_ip_j \in E(DG)$. Thus the point $p_k \in R(p_i)$ must have a Gabriel edge with p_i. Similarly, if $p_k \notin R(p_i)$, then there must be at least one $R(p_i, \widehat{p_j})$ such that $p_k \in R(p_i, \widehat{p_j})$ and $p_ip_j \in E(DG)$. Thus the edge $p_ip_k \notin E(GG)$ as shown in Lemma 9 which completes the proof of this lemma. □

Lemma 13. *The union of the regions* $R(p_i)$ *for all* $p_i \in V(DG)$ *is the plane* $\mathbb{R}^2$ *i.e.,*

$$\bigcup_{p_i \in V(DG)} R(p_i) = \mathbb{R}^2$$

Proof. Suppose for the purpose contradiction that the union of the regions $R(p_i)$ for all $p_i \in V(DG)$ cannot construct the plane $\mathbb{R}^2$. Thus there exists a region h such that $h \subseteq \mathbb{R}^2$ and $h \nsubseteq \bigcup_{p_i \in V(DG)} R(p_i)$. Thus any point p_k inserted into the region h cannot form an edge $p_ip_k \in E(GG)$ for all $p_i \in V(GG)$ according to Lemma 12. Thus p_k remains as an isolated vertex in GG which contradicts the Lemma 6. □

5.3 Effective Cell

An *effective cell* is the non empty region $h \subset \mathbb{R}^2$ induced by the common intersection of the each unique region r such that $r \in R_\pi(p_i)$ and $h \subseteq r$ for every $p_i \in V(DG)$. Formally, *effective cell* ec can be defined as

$$\begin{aligned} ec = \bigcap \{r : \forall p_i (p_i \in V(DG) \\ \implies \exists! r (r \in R_\pi(p_i) \wedge h \subseteq r))\} \end{aligned}$$

where ec cannot be empty. These bounded and unbounded *effective cell* joined together to form the cell complex which can be constructed using the intersection of the regions $R(p_i)$ for all $p_i \in E(DG)$.

Theorem 6. *The set of adjacent vertices for an insertion point is unique in an* effective cell*.*

Proof. Let an *effective cell* ec of region $h \subset \mathbb{R}^2$ be the common non empty intersection of elements of R which can be written as

$$\begin{aligned} R = \{r : & \forall p_i(p_i \in V(DG) \\ & \implies \exists! r(r \in R_\pi(p_i) \wedge h \subseteq r))\} \end{aligned}$$

Let $R_m \subseteq R$ be defined as follows

$$R_m = R \cap \{R(p_i) : p_i \in V(DG)\}$$

According to Lemma 12 a point p_j is Gabriel adjacent to all p_i *iff* $p_j \in R(p_i)$. Thus the set of adjacent vertices V_a can be found as

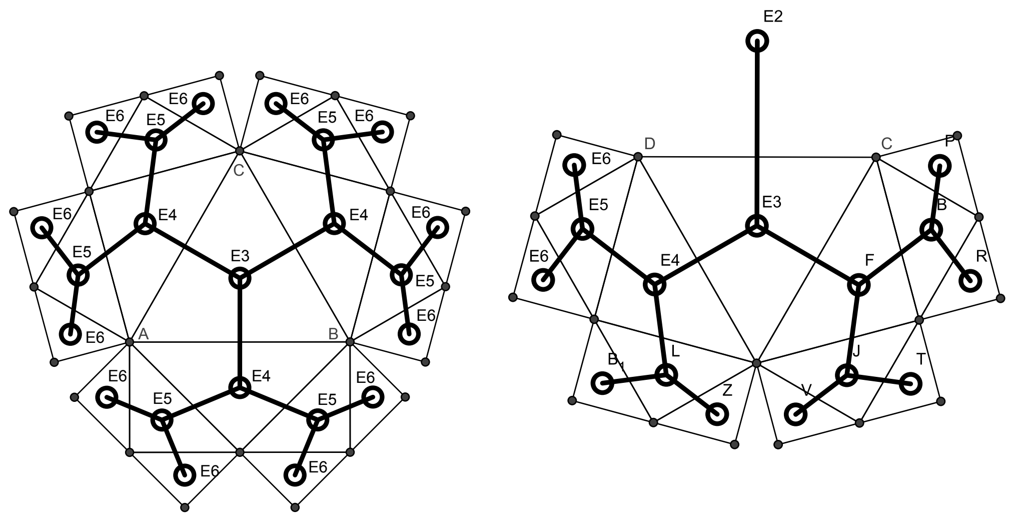
$$V_a = \{p_i : R(p_i) \in R_m\}$$

Thus the set of Gabriel adjacent vertices V_a remain unique throughout the *effective cell* ec since V_a depends on R_m which is a subset of R. □

Theorem 7. *The expected number of effective cell in a Delaunay graph is linear.*

Proof. It is sufficient to prove that the upper bound on the expected number of effective cell in a Delaunay graph DG with n vertices is linear. Suppose a random variable X represent the degree of a vertex on the probability space $V(DG)$ and X_i is the degree of ith vertex in the graph, DG. Thus the expected degree $E[X]$ of a vertex in the DG is

$$\begin{aligned} E[X] &= E\left[\sum_{i=1}^{n} X_i\right] && (5.3.1) \\ &= E\left[2(3n-3)\right] && (5.3.2) \\ &= \frac{6n-6}{n} < 6 && (5.3.3) \end{aligned}$$



(a) All possible adjacent (b) On the convex hull

Figure 5.5: Average number of effective cell is linear

where the graph DG can have at most $3n - 3$ edges. Let p_i be a new point inserted into the triangle ABC of DG graph as shown in Figure 5.5a. There is a set S of twenty four points in the graph DG which are the all possible points for constructing six expected edges incident with p_i at one end. The point p_i cannot be adjacent with other points except the points $p_j \in S$ through the expected edges. Since the Gabriel graph is a subgraph of the Delaunay graph, the expected Gabriel edges must be the subset of the six expected Delaunay edges incident with p_i. Therefore, the effective regions $R(p_j)$ for all $p_j \in S$ are the all possible effective regions which can be intersected inside the triangle ABC to construct the expected Gabriel edges. Since any two effective regions can intersect only at two points, the total number of intersection points inside the triangle ABC is $24(24 - 1)$. The total number of vertices of all the effective regions in the graph DG is $2(3n - 3)$ since the graph DG can contain maximum $3n - 3$ edges. Thus the total number of vertices of the planar graph RG constructed by the effective regions of the graph DG inside the convex hull CH of vertices $V(DG)$ is $24(24 - 1)(2n - 2) + 2(3n - 3)$ since the graph DG can have at most $2n - 2$ triangles. Although many intersection points are coincide with other intersection points, for upper bound estimation we consider all intersection points. Let B be the set of points on the boundary of the convex hull CH. The regions $R(\widehat{p_i}, \widehat{p_j})$ for all $p_i, p_j \in B$ are not intersect with each other outside the convex hull CH since each angle between the edges of the convex hull CH is convex. If any new point p_k inserted into the region $R(\widehat{p_i}, \widehat{p_j})$, then there is a set S of seventeen points in the graph DG which are the all possible points for constructing six expected edges as shown in Figure 5.5b. Thus the number of intersection points inside the region $R(\widehat{p_i}, \widehat{p_j})$ is $17(17-1)$ and there could be maximum n edges on convex hull. Thus the total intersection point outside the convex hull is $17(17-1)n$. Thus total intersection point of RG is $24(24-1)(2n-2) + 2(3n-3) + 17(17-1)n$. Thus from Corollary 2, the maximum number of bounded faces (i.e., effective cells) is $2764(n-1) + 543 - 1$. A connected simple planar graph has only one unbounded face but we can have maximum $2n$ unbounded face. Thus total number of bounded and unbounded effective cell is $2766(n-1) + 544$ which is $O(n)$. $\square$

Visitor Query: An application

Let us consider a region with n number of P2P service stations A where each $p_i \in A$ can provide the service to a client p_k if there is no other service stations inside the circle p_ip_k. Two service stations p_i and p_j can also provide service to each other if no other service stations or client is inside the circle p_ip_j. A visitor wants to visit one specific place in the region to cover as many service stations as possible. So, the query of the visitor is to get a specific subregion where he can connect with maximum number of service station. Certainly, the connectivity between the service stations follows the Gabriel edge. According to our spatial subdivision, each *effective cell* has a direct mapping with a set of points A in the Gabriel graph, i.e. if a new point p_k is inserted into anywhere in the *effective cell* then there must create edge p_ip_k for every p_i in A. Eventually, if the visitor p_k wants to move further out of *effective cell*, he can choose another *effective cell* adjacent to his current *effective cell* based on the maximization of service coverage.

Lemma 14. *The construction of* effective region *of a vertex takes $O(M \log M) + O(M)$ running time.*

Proof. Let $S = \{s_1, s_2, s_3, \ldots, s_M\} \subseteq V(GG)$ is the set of adjacent point of $p_j \in V(GG)$ and $s_ip_j \in E(DG)$ where $1 \le i \le M$. We assume that the set S is sorted by the anticlockwise order around p_j and p_js_m has the angular adjacency with p_js_1. Now if we want to compute the cut of $L(\widehat{s_i}, p_j)$ we have to consider the $L(\widehat{s_{i+1}}, p_j)$ and $L(\widehat{s_{i-1}}, p_j)$ because s_{i+1} and s_{i-1} are the adjacent of p_j and those have minimum anticlockwise and clockwise angle respectively with respect to s_ip_j. Similarly $L(\widehat{s_{i+1}}, p_j)$ is cut by $L(\widehat{s_{i+2}}, p_j)$ and $L(\widehat{s_i}, p_j)$. So, including the cost of sorting of M points, the cut of the perpendicular line of every Delaunay edge connected with p_j can be computed in $O(M \log M) + O(M)$ time. □

5.4 Dynamic Update

The goal of this section is to provide the technique of incremental update *effective region* for each insertion and deletion of point into the GG. Let us consider an *effective region* diagram where all points p_r adjacent to p_i in the DG are ordered by clockwise angular distance around p_i. If a new point p_k is inserted into the intersection of *effective regions* of m number of p_i then each p_ip_k must be the edge in new GG. To update the graph, the following three things should be taken into consideration: Firstly, p_k may fell inside the circle of a set of pairs of points where each pairs have edges in GG. So, all these edges must be deleted. Fortunately, this can be solved automatically during the update of Delaunay edges. Secondly, for the insertion of p_k, a number of Delaunay edges that are associated to p_k will be inserted and few edges that are already exists will be deleted. Moreover, if a point become the Delaunay neighbour of p_k then the angularly sorted list around p_k needs to be updated. Finally, after getting the updated Delaunay graph, the reconstruction of all the associated *effective regions* must be done.

On the other hand, if a point p_k is deleted from the graph then a star shaped polygon associated to p_k are re-triangulated. During the re-triangulation process, all the angularly sorted lists and effective regions must be updated exactly the opposite way followed in insertion.

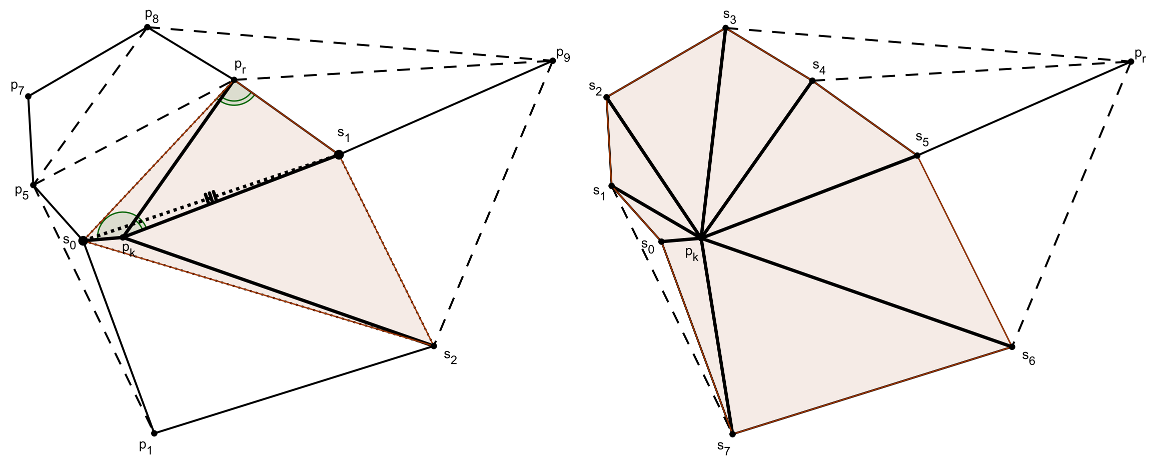
5.4.1 Insertion

Each of the *effective region* of p_i can be partitioned into $M(p_i)$ number of triangular wedges around each $p_i \in V(GG)$ by each $p_i p_r$. Since each p_r is sorted in clockwise angular distance around p_i, and so each wedges are also naturally kept by the same order $p_r p_i p_{r+1}$. If a point is inserted into the intersection of the *effective regions* of some points, following operations are needed to be done:

1. Update the Delaunay graph
2. Update the angular order for each of the affected points
3. Recompute the *effective regions* for each of the affected points

Update the Delaunay and Gabriel edge

Let p_k is inserted into the intersection of a clockwise angular sorted set of points $S = \{s_0, s_1, s_2, \dots\}$ of length $m(p_k)$ where $m(p_k)$ is the degree of p_k in GG. We consider a virtual polygon by S around p_k where the polygon S can either be bounded or unbounded. In each of the triangular wedges $\Delta s_i p_k s_{i+1}$, the edge $p_k s_i$ and $p_k s_{i+1}$ must exists in the updated graph because p_k is inside the *effective region* of both s_i and s_{i+1} [Lemma 12]. The edge $s_i s_{i+1}$ be clearly the candidate for flipping with $p_k p_r$ where p_r is opposite to the p_k in the quadrilateral $s_i p_k s_{i+1} p_r$ [see Figure 5.6a]. Now, if the circle $s_i p_k s_{i+1}$ is empty then $s_i s_{i+1}$ must be an edge in Delaunay graph and since both the triangle of the diagonal



(a) Insertion of p_k (b) Updated Delaunay graph for inserting p_k

Figure 5.6: Updating *effective region*, $R(p_k)$

s_is_{i+1} remain unchanged, no further check is necessary along that triangles and the next triangular wedge $\Delta s_{i+1}p_ks_{i+2}$ is picked to check. However, if s_is_{i+1} is flipped into p_kp_r then we have to consider another two triangles $s_ip_kp_r$ and $p_rp_ks_{i+1}$ to check exactly the same way until we get no flipping edge [see Figure 5.6b].

One exception may arise when $\angle s_ip_ks_{i+1}$ is not convex [see Figure 5.7]. So, we do not have any valid triangle named $s_ip_ks_{i+1}$. Fortunately, there is an easy solution of this exception case. if $\angle s_ip_ks_{i+1}$ slightly less than $180°$ then the large circle cover almost the half space of s_is_{i+1}. We can say, if a point p_r is left to the line s_ip_k then there must be a Delaunay edge between p_k and p_r. If there are multiple points p_r at the left side of s_i, we take such a p_r which has the minimum angular distance with s_ip_k and check again whether $\angle s_ip_kp_r$ is still concave. In this case, the same process is followed until it is convex. After obtaining the convex angle, the previous techniques can be applied directly. However, if no such p_r is found at the left of s_i then we can skip the wedge (triangle) $s_ip_ks_{i+1}$.

It is important fact that each time of flipping of edge, a vertex p_r will be the adjacent to p_k therefore, we get a star shaped polygon centered with p_k [see

Figure 5.6b]. This gives us the computational cost is proportional to the degree of p_k, say $M(p_k)$in Delaunay graph. We can distinguish between Delaunay and Gabriel edge by a circle test in constant time.

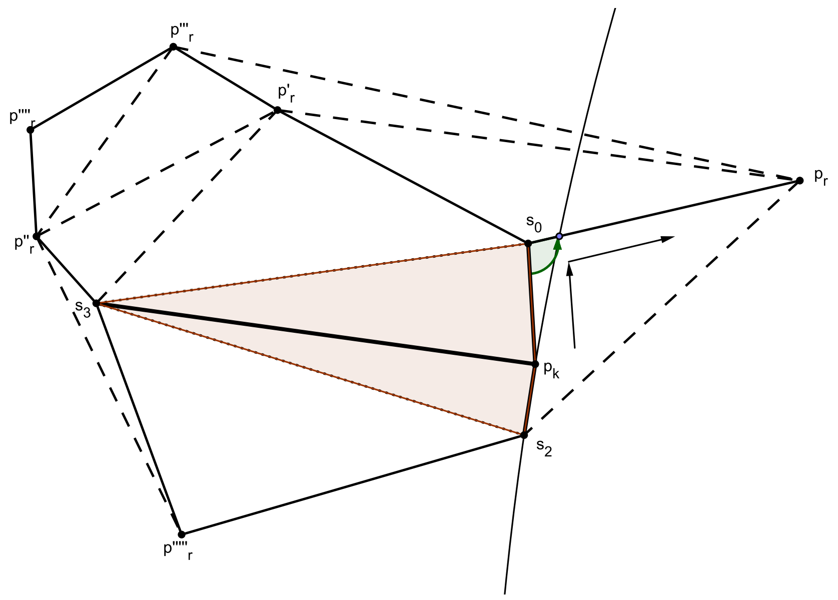


Figure 5.7: Concave angle for p_k

Update the angular order

Angular order is necessary for the computation of *effective region* very efficiently. So, in each steps of the update of Delaunay graph we need to update the clockwise ordering of neighbour of each point.

Let us consider that each $s_i \in S$ is sorted around p_k by clockwise angular distance and we start from the triangular wedges $s_1 p_k s_2$. If p_k obtain a new neighbour p_r then p_r can be inserted into the sorted list around p_k that can be done in constant time. p_k will also be the neighbour of p_r but it needs a binary search on $O(log(M(p_r)))$ to find the appropriate position of p_k from the sorted list around p_r. Similarly, the ordering of neighbour around both s_1 and s_2 need to be updated that can be done in $O(log(M(s_1)))$ and $O(log(M(s_2)))$ respectively. After completing all the necessary updates, total costs for updating the angular information can be $O(M(p_k) \log M(p_k))$ because we need to update the angular order of all Delaunay adjacent around p_k.

Update of effective region of each p_k

From Lemma 14 we know that if the neighbours of p_k are in ordered by clockwise angular distance around p_k then it takes $M(p_k)$ times to compute all the *effective regions* and therefore $O(M(p_k))$ number of test is sufficient for update also.

In Algorithm 5, $IN(a, b, c, d)$ is the function for testing whether d is inside the circle abc or not. $CCW(abc)$ refers to the counter-clockwise angle by a, b and c. One additional variable named $coverboth$ is used to check whether both $p_k s_i$ and $p_k s_{i+1}$ is checked one by one until $s_i p_k s_{i+1}$ become convex.

Algorithm 4 SwapTest(s_i, s_j, p_k, p_r)

```
BEGIN
if IN($s_i, s_j, p_k, p_r$) then
   flip $s_i s_j$ by $p_k p_r$
   update the neighbouring order of and around $p_k$, $p_r$, $s_i$ and $s_j$
   SwapTest($s_i, p_k, p_r, p'_r$)
   SwapTest($s_j, p_k, p_r, p''_r$)
else
   $L(i, k) \leftarrow$ perpendicular bisector line of $s_i p_k$
   $L(j, k) \leftarrow$ perpendicular bisector line of $s_j p_k$
   $C$ = intersection($L(i, k)$, $L(j, k)$)
   if $C$ is opposite side of $p_k$ by $s_i s_j$ then
      mark $s_i s_j$ as Gabriel edge
   end if
end if
END
```

5.4.2 Delete

To update the graph after the deletion of a point p_k all the sorted order can be updated total $O(M(p_k))$ time which is exactly similar to the insertion. Now, we have to estimate the update cost of deletion. We know that if a set of point S is incident to p_k and p_k is deleted, then no triangle get affected outside the convex polygon by S. Aggarwal, Guibas, Shaxe and Shor [32] provides an linear time algorithm for finding the Delaunay triangulation of a convex polygon. Since, size of S is the number of adjacent of p_k, we can update the DG and all associated angular order by $M(p_k)\log M(p_k)$ times.

Algorithm 5 Insertion of p_k

```
BEGIN
$S = \{s_0, s_1, s_2, \ldots, s_{m-1}\}$ {$m$ = degree of $p_k$ in $GG$}
for $i = 0 \to m-1$ do
  $j \leftarrow (i + 1( \mod m$
  if $\angle s_i p_k p_j$ is convex then
    $p_r \leftarrow$ vertex on the opposite sharing triangle of $p_i p_j$
    SwapTest($s_i, s_j, p_k, p_r$)
  else
    $a = s_i$, $c = s_j$, $coverboth \leftarrow 2$
    while $\angle a p_k c$ is concave do
      $b = \{p \in V(DG) : ap \in V(DG)$ and $CCW(p_k a p)$ is minimum $\}$
      if $b$ does not exists then
        $coverboth \leftarrow coverboth - 1$
        if $coverboth = 0$ then
          *break*
        end if
        $swap(a, c)$
        *continue*
      end if
      connect $p_k b$
      update the neighbouring order of and around $p_k$ and $b$
      $a = b$
    end while
  end if
end for
END
```

Theorem 8. *The update algorithm for insertion and deletion of a point in the intersection of a set of* effective regions *takes $O(M \log M)$ times*

Proof. If a point p_k is inserted into the intersection of some effective regions then we have three steps to follow to update each of the effective regions. The time for update of angular adjacency takes the dominating cost $M(p_k)\log M(p_k)$ and both the update of Delaunay graph and computing the *effective region* takes only $M(p_k)$ times. So, it takes total $M(p_k)\log M(p_k) + M(p_k) = O(M \log M)$ time. We use M rather than $M(p_k)$ for generalization. The discussion on deletion directly tells that the cost of update for deletion for any point also takes $O(M \log M)$ □

Theorem 9. *The query for the existence of a Gabriel edge can be answered in* $O(\log(M))$ *time and including the update cost it takes total* $O(M \log(M))$ *times.*

Proof. From Lemma 12, we proved that an edge p_ip_j exists in $E(GG)$ *iff* a point p_j is inserted into the *effective region* $R(p_i)$. So, all we need to know whether p_j is inside the *effective region* $R(p_i)$. The best known algorithm for querying whether a point is inside the polygon takes $O(\log n)$ time [33] where n is the number of vertices in the polygon. Since, the number of vertices in the *effective region* $R(p_i)$ is $M(p_i)$, it takes $O(\log(M(p_i)))$ times to check whether the circle p_ip_j is empty or not. In general, for any point p_i the cost can be written as $O(\log(M))$.

On the other hand, for each movement of points the *effective region* of p_i can be changed. In Theorem 8, we established that it takes $O(M \log(M))$ times for updating all the *effective region* in Gabriel graph if a new point is inserted or deleted. So, for the movement of point each time, it takes total $O(M \log(M)) + O(\log(M)) = O(M \log(M))$ times. □

Chapter 6

Results and Discussion

In this thesis, we provide the definition of *view* and propose a solution for maintaining the *view* dynamically. The framework provided by Guibas et al [6] for maintaining a dynamic scenario has been used to measure the performance in terms of efficiency, locality and responsiveness. In our algorithm, we achieve the locality $O(1)$, responsiveness $O(M \log(M))$ and efficiency $O(M \log(M)\lambda_s(n^2))$ where n is the number of points in projection plane and M is the degree of any vertex in the Delaunay graph.

To obtain $O(M \log(M))$ as the response time for the dynamic update of *view*, we propose a technique for spatial sub-division of Gabriel graph which also has a wide range of applications in various research areas like geographical variation analysis, least square error analysis, mobile facility location, wireless sensor network, gene sample clustering and so on. We also provide an algorithm for maintaining the Gabriel graph dynamically with $O(M \log(M))$ running time.

Since we find the computational cost for updating our proposed effective region for insertion and deletion is related to M and m, we need to know its growth. Although in worst case, $m = M = n$, gives us the $O(n \log n)$ time for update. However, expected and amortize analysis gives that both m and M are constant for sequence of insertion and deletion.

Lemma 15 ([18]). *Let GG be the Gabriel graph formed on $n \geq 3$ vertices that are placed in a uniformly random manner on the unit square. Then the expected degree of a particular point $v \in GG$ is 4.*

On the other hand, according to Euler's theorem, the expected degree of a vertex in a planar graph is less than six. Kao et. al [29] shows that for many natural point distribution, the maximum degree in a Delaunay graph is rarely exceeds 16.

We show that the order of the growth of the number of edges in Gabriel graph remain amortized constant in the s-sequence of insertions of points. In potential (or physicist's) method, it is derived a potential function characterizing the amount of extra opportunity to get the maximum edge in the graph in each step. This potential either increases or decreases with each successive insertion, but cannot be negative.

During the computation of upper bound of the number of edges we get two different scenarios. Firstly, if we always try to get the maximal edge Gabriel graph for each insertion, either two or three edges can be added. Secondly, we can place a vertex into the n-cycle. In this case, it is possible to get additional n number of new edges for one insertion. However, to get such an n-cycle in the earlier steps, the potentiality of the whole graph must be increased because this is not the maximal edge graph. It is an important fact that after the insertion of the vertex into the n-cycle the potential of the graph is dropped dramatically into two. So, the number of edges in amortize case is $n + (2 - n) = 2$. So, if we sum up the amortize time for all the sequence of insertion and computed then the number of edges in GG is surely amortized constant.

Similarly, for Delaunay graph, we know from Euler theory that the sum of degrees is the twice the number of edges in the graphs. Therefore,

$$\sum_{1 \le i \le n} M(p_i) = 2(3n - 6) < 6n \tag{6.0.1}$$

So, the amortize size of the degree of any vertex is $\frac{6n}{n} = 6$ which is constant.

The proposed spatial subdivision is advantageous for rapid movement of points. The unit movement of a point is nothing but the three ordered operations: deletion, changing the unit position and then insertion into the changed position. Let us have a finite $m \times m$ square grid space where there are m^2 number of discrete position for a point. This is actually similar to the computer graphics with a specific resolutions. Now, among n number of points from P in the Gabriel graph, the task of $p_i \in P$ is to traverse all the position (or pixels) of the space once at a time. It is provided that all other point except p_i are fixed. The update of the Delaunay graph require $O(n)$ time in worst case but $O(\log n)$ in amortize case. So after completing the tour, it costs $O(m^2 \log n)$ time for updating the Delaunay graph for each of the movement p_i.

From Theorem 6, we know that any point in an effective cell has the same set of Gabriel edge as its adjacent. So, until leaving the effective cell it is unnecessary to query for Gabriel adjacency. Although we did not compute the bound of number of effective cell, we can ensure that the number will not greater than the number of sparse cell. Since the sparse cell is the arrangement of perpendicular lines of each edges in the Delaunay graph, the number of effective cell can be bounded by $O(n^2)$ where n is the number of vertices in the Gabriel graph. So, the average number of grid points(pixels) per cell is $\frac{m^2}{n^2} = (\frac{m}{n})^2$. In most of the practical case m is much larger than n and hence, each cell contains a significant amount of positions(pixels). Therefore, it is sufficient to make query only $(\frac{m}{n})^2$ number of times to know about the adjacent edges in GG. So, if we take the advantages of spatial subdivision, its takes $\frac{m^2 \log n}{n^2}$ times for the complete traversal in the grid. This amount certainly n^2 times less costly than the previous one.

Our different spatial subdivision methods based on Gabriel adjacency relationship such as sparse cell, effective region and effective cell provide different means to determine the Gabriel adjacent in the regions. Each subdivision of the plane provides a consistent adjacency relationship based on some predefine criteria. One of the main advantages of these subdivisions is that we consider a set of points together instead of considering each geographical point separately. We can quickly store and retrieve valuable information related to the adjacency relationship including gene flow and other genetic associations like gene disorder for some specific geographical areas. This work also gives the answer for the same types of queries in facility location, wireless sensor network, Data similarity and error analysis etc when the connectivity is given by Gabriel graph. For instance, one may want to place a node in the wireless sensor network such that the connectivity of the graphs needs to be updated only in a very small portion of the whole graph. Similarly, a client may find a location for getting maximum as well as minimum facility node (shopping mall, mobile station etc.). Biologist may want to find a common region with multiple ethnic groups and also want to analysis the spread of a epidemic deceases based on the connectivity between each ethnic groups. There are many other additional advantages of our proposed effective cell. A short outline is given below:

- *Empty Disk*: One of the variations of Empty Disk problem is the adaptation of a constraint that no vertex in any empty disk can collide with other empty circle. A common question may arise: to make maximum number of empty disk where we should insert a new point into the empty disk graph.

- *Least Square Adjacency*: From the alternative definition of Gabriel graph tells us that two points are least square adjacent if there is an edge in the corresponding Gabriel graph. If a set of data is clustered based on Gabriel graph and we want to know the region based on the maximum similarity and also dissimilarity of a test data with the other sample data. Our effective cell gives the answer quite efficiently. Similarly, we can also find the minimum and maximum least square errors in a test data with the sample data.

- *Gene Sampling*: Biologist are very interested in biological models for gene flow that use Gabriel connectivity among sampling station points. In some epidemic deceases, biologists want to analyze the possible transmission of that deceases from one region to another region based on the regional contiguity among different localities. If the regional contiguity is defined by the Gabriel graph then our proposed spatial subdivision give the answer of many important queries like most threatening area and safest area from the particular epidemic deceases.

- *Geographic Variation Analysis*: The uses of Gabriel graph in the clustering of Geographic regions are very popular among GIS scientists. Their common query is to find the region with maximally as well as minimally diverse ethnic group.

Chapter 7

Conclusion

7.1 Conclusion

In this research, if two projected points cross each other along any of the axes and the circle that is constructed with the diameter connecting those two points contains no other points then before and after such an event, the configurations have been considered as two different *views*. At each of the topological event, the term *view* is updated by maintaining the order of points in the projection plane. Along with the technique for efficient query of the emptiness of a circle, the necessary data structures and algorithms have been provided for maintaining the attribute *view* dynamically with responsiveness $O(M \log{(M)})$ and efficiency $O(M \log{(M)} \lambda_s(n^2)))$. Additional contribution of this paper is the technique of spatial subdivision of Gabriel graph that provide a new direction for the dynamic maintenance of Gabriel graph. Moreover, the concept of effective regions and effective cells are very useful in many other proximity problems that uses Gabriel graph.

7.2 Future Study

Despite that there is no a complete agreement on the definition of *view* in 3D, we believe our new definition and its dynamic maintenance will be helpful for future research. However, few more problems related to the attribute *view* remain open for future study. For example,

- If we consider the circular ordering for 3D point set, we do not find any projection place if the view point is inside the convex polyhedra of the point set. Rather the points are projected on a sphere and the circular order on the sphere is undefined.
- Computing the visibility complex is the process of computing the time of topological events based on the trajectory of movement of view point. It will be a very challenging as well as attractive task to find the the visibility complex for the attribute *view*.
- The spatial sub-division of Gabriel graph motivates us to obtain the same type of sub-division of Delaunay graph as well.
- Since, Minimum Spanning Tree (MST) is the subgraph of Gabriel graph, we can apply the similar techniques of spatial subdivision to get an efficient maintenance of MST.

Bibliography

[1] O. Devillers, V. Dujmovic, H. Everett, S. Hornus, S. Whitesides, and S. Wismath, “Maintaining visibility information of planar point sets with a moving viewpoint,” *International Journal of Computational Geometry and Applications*, vol. 57, no. 4, pp. 297–304, 2007.

[2] S. Ghali and J. Stewart, “Incremental update of the visibility map as seen by a moving viewpoint in two dimensions,” in *Proceedings of the Seventh International Eurographics Workshop on Computer Animation and Simulation*, pp. 1–11, 1996.

[3] S. Ghali and J. Stewart, “Maintenance of the set of segments visible from a moving viewpoint in two dimensions,” in *Proceedings of the 12th Annual ACM Symposium on Computational Geometry (SCG 96)*, pp. V3–V4, 1996.

[4] M. W. Bern, D. P. Dobkin, D. Eppstein, and R. L. Grossman, “Visibility with a moving point of view,” *Algorithmica*, vol. 11, no. 4, pp. 360–378, 1994.

[5] H. Akbari and M. Ghodsi, “Visibility maintenance of a moving segment observer inside polygons with holes,” in *Proceedings of the 22nd Canadian Conference on Computational Geometry (CCCG 2010)*, (Winnipeg MB, Canada), pp. 117–120, 2010.

[6] L. J. Guibas, *Handbook of Data Structures and Applications*, ch. 23, pp. 1–18. Chapman and Hall/CRC, 2004.

[7] H. Xiao, *Kinetic Visibility*. PhD thesis, Queen's University Kingston, School of Computing, Ontario, Canada K7L 3N6, 2007.

[8] H. Plantinga and C. R. Dyer, "Visibility, occlusion, and the aspect graph," *International Journal on Computer Vision, Springer Netherlands*, vol. 5, no. 2, pp. 137–160, 1990.

[9] J. Basch, L. J. Guibas, C. D. Silverstein, and L. Zhang, "A practical evaluation of kinetic data structures," in *Proceedings of the 13th Annual Symposium on Computational Geometry (SCG 97)*, (ACM, NY, USA), pp. 388–390, 1997.

[10] J. Basch, H. Devarajan, P. Indyk, and L. Zhang, "Probabilistic analysis for discrete attributes of moving points," *International Journal of Computational Geometry and Applications*, vol. 13, no. 1, pp. 5–22, 2003.

[11] J. Basch, J. Erickson, L. J. Guibas, J. Hershberger, and L. Zhang, "Kinetic collision detection between two simple polygons," *Computational Geometry: Theory and Application, Elsevier*, vol. 27, no. 3, pp. 211–235, 2004.

[12] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Zoology (Society of Systematic Biologists)*, vol. 18, no. 3, pp. 259–270, 1969.

[13] J. Choo, R. Jiamthapthaksin, C. Chen, O. U. Celepcikay, C. Giusti, and C. F. Eick, "Mosaic: A proximity graph approach for agglomerative clustering," in *Proceedings of the 9th International Conference on Data Warehousing and Knowledge Discovery*, pp. 231–240, 2007.

[14] O. Pujol and D. Masip, "Geometry-based ensembles: Toward a structural characterization of the classification boundary," *IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Computer Society*, vol. 31, pp. 1140–1146, 2009.

[15] M. Aupetit, "High-dimensional labeled data analysis with gabriel graphs," in *Proceedings of the European Symposium on Artificial Neural Networks, d-slide,* (Bruges, Belgium), pp. 21–26, 2003.

[16] W. Zhang and I. King, "A study of the relationship between support vector machine and gabriel graph," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN 02), IEEE,* (Honolulu, HI, USA), pp. 239–244, 2002.

[17] D. Laloe, K. M. Goudarzi, J. A. Lenstra, P. A. Marsan, P. Azor, R. Baumang, D. G. Bradley, M. W. Bruford, J. Canon, G. Dolf, S. Dunner, G. Erhardt, G. Hewitt, J. Kantanen, G. O. Ruff, I. Olsaker, C. Rodellar, A. Valentini, and P. Wiener, "Spatial trends of genetic variation of domestic ruminants in europe," *European Cattle Genetic Diversity Consortium, Econogene Consortium. -In: Diversity 2010*, vol. 2, no. 6, pp. 932–945, 2010.

[18] D. W. Matula and R. R. Sokal, "Properties of gabriel graphs relevant to geographic variation research and clustering of points in the plane," *Geographical Analysis*, vol. 12, no. 3, pp. 205–222, 1980.

[19] S. E. Howe, *Estimating Regions and Clustering Spatial Data: Analysis and Implementation of Methods Using the Voronoi Diagram*. PhD thesis, Brown University, Providence, USA, 1978.

[20] W. Wang, "The spanning ratio of β-skeletons," in *Proceedings of the 15th Canadian Conference on Computational Geometry (CCCG 2003)*, (Halifax, Nova Scotia), pp. 35–38, 2003.

[21] P. Bose, J. Cardinal, S. Collette, E. D. Demaine, B. Palop, P. Taslakian, and N. Zeh, "Relaxed gabriel graphs," in *Proceedings of the 21st Canadian Conference on Computational Geometry*, (Winnipeg MB, Canada), pp. 169–172, 2009.

[22] P. Bose, J. Cardinal, S. Collette, F. Hurtado, M. Korman, S. Langerman, V. Sacristan, and M. Saumell, "Some properties of higher order delaunay and gabriel graphs," in *Proceedings of the 22nd Canadian Conference on Computational Geometry (CCCG 2010)*, (Winnipeg MB, Canada), pp. 13–16, 2010.

[23] J. Basch, L. J. Guibas, and J. Hershberger, "Data structures for mobile data," *Journal of Algorithm*, vol. 31, no. 1, pp. 1–28, 1999.

[24] M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications, Cambridge University Press*. 1995.

[25] S. Hart and M. Sharir, "Nonlinearity of davenport schinzel sequences and of generalized path compression schemes," *Combinatorica*, vol. 6, pp. 151–177, 1986.

[26] P. K. Agarwal, M. Sharir, and P. Shor, "Sharp upper and lower bounds on the length of general davenport schinzel sequences," *Journal of Combinatorial Theory Ser. A*, vol. 52, pp. 228–274, 1989.

[27] M. Sharir, "Improved lower bounds on the length of davenport schinzel sequences," *Combinatorica*, vol. 8, pp. 117–124, 1988.

[28] Wikipedia, "Inverse ackermann function." `https://secure.wikimedia.org/wikipedia/en/wiki/Ackermann_function`, Retrieved on January 2011.

[29] T. Kao, D. M. Mount, and A. Saalfeld, "Dynamic maintenance of delaunay triangulations," in *Proceedings of Auto-Carto*, (College Park, MD, USA), pp. 219–233, 1991.

[30] L. Guibas, D. Knuth, and M. Sharir, "Randomized incremental construction of delaunay and voronoi diagram," in *Proceedings of International Col-*

loquium on Automata, Languages and Programming (ICALP), pp. 414–431, 1990.

[31] M. D. Berg, M. V. Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry Algorithms and Applications*, ch. 9, pp. 183–210. Springer, second ed., 2000.

[32] A. Aggarwal, L. J. Guibas, J. Saxe, and P. W. Shor, "A linear-time algorithm for computing the voronoi diagram of a convex polygon," *Discrete and Computational Geometry*, vol. 4, pp. 591–604, 1989.

[33] E. Haines, "Point in polygon strategies," *Graphics Gems*, vol. 4, pp. 24–46, 1994.