M.Sc. Engineering Thesis

# Voronoi Neighbors: Optimization, Variation and Games

by
Md. Muhibur Rasheed
Student No. 100505018P

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 (CSE)
Bangladesh University of Engineering and Technology (BUET)
Dhaka–1000, Bangladesh

July 21, 2008

The thesis titled "Voronoi Neighbors: Optimization, Variation and Games", submitted by Md. Muhibur Rasheed, Student No. 100505018P, Session October 2005, 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 was held on July 21, 2008.

| | | |
|---|---|---|
| 1. | [illegible]<br>Dr. Masud Hasan<br>Assistant Professor<br>Department of CSE, BUET, Dhaka | Chairman<br>(Supervisor) |
| 2. | [illegible]<br>Dr. Md. Saidur Rahman<br>Professor and Head<br>Department of CSE, BUET, Dhaka | Member<br>(Ex-officio) |
| 3. | [illegible]<br>Dr. Mahmuda Naznin<br>Assistant Professor<br>Department of CSE, BUET, Dhaka | Member |
| 4. | [illegible]<br>Dr. A.K.M. Ashikur Rahman<br>Assistant Professor<br>Department of CSE, BUET, Dhaka | Member |
| 5. | [illegible]<br>Dr. Ashfaqur Rahman<br>Assistant Professor<br>Department of Computer Science<br>American International University, Bangladesh | Member<br>(External) |

[illegible]

# Candidate's Declaration

This is to certify that the work entitled "Voronoi Neighbors: Optimization, Variation and Games" is the outcome of the investigation carried out by me under the supervision of Dr. Md. Masud Hasan in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka - 1000, Bangladesh. 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.

[illegible]

Md. Muhibur Rasheed

# Acknowledgement

All praises due to Allah, Lord of the Worlds, who granted me the ability to finish this thesis.

I would like to express my heartfelt gratitude to Dr. Masud Hasan, my thesis supervisor, for his continuous patience, guidance and encouragement. It was really a honour for me to have the opportunity to work with such a great researcher like him. Most of the works of this thesis were the results of joint work with him.

I am also thankful to the members of board of examiners - Dr. Md. Saidur Rahman, Dr. Mahmuda Naznin, Dr. A.K.M. Ashikur Rahman and Dr. Ashfaqur Rahman - for their valuable suggestions, advice and correction.

# Abstract

Recently several researchers have formulated the competitive facility location problem through Voronoi diagrams where a new site is placed in such a location that its Voronoi region is maximized. One notable result gives a randomized approximation algorithm that works in general settings and another gives a deterministic algorithm that only works in very restricted settings. Research is also going on to find winning strategies for Voronoi games based on area maximization. So far, winning strategies could be found for 1 round games in 2D and for $n$ round games in 1D.

This thesis addresses cooperative facility location which can be modeled through Voronoi neighborships. We developed a solution which, given $n$ points in 2D, finds the location where a new site should be placed so that it gets the maximum or the minimum number of the existing sites as its neighbors. We also developed a solution to the problem of finding the optimum number of new sites that need to be added to get all existing sites as neighbors. We analyzed optimal playing strategies for a game where the basic objective is to acquire more neighbors than the opponent. Several variations of the game were considered and we gave winning strategies for some variations, for some variations we showed that it always ends in a tie, and for other variations we showed that the game ends in a tie unless a player plays non-optimally. We also developed an algorithm to generate the arrangement of Delaunay circles in $O(n)$ time, and detect all intersections of circles within two Voronoi layers.

# Contents

# List of Figures

# Chapter 1

# Introduction

Imagine that you are the owner of a large shopping chain and you have multiple shops situated in different locations (for example, in Figure 1.1(a) some shops are shown (solid circles)). Now, you want to set up warehouses in convenient locations to support the smooth flow of commodity to these shops. Though it would be most convenient to set up one warehouse near every single shop (as shown in Figure 1.1(b) (warehouses are represented by solid rectangles)), but there are some constraining factors that you have to consider like the cost of setting up the warehouses and the cost of maintaining them. On the other extreme, if you set up only a few warehouse (as shown in Figure 1.1(c)), then both the cost and time required for transportation would rise. Therefore, the objective is to find the exact number of warehouses and the exact locations of them which ensures that the total cost is minimized and the total profit is maximized (Figure 1.1(d) proposes a possibly good trade-off).

This problem is popularly known as *facility location problem*. In general, there are a set of facility receivers (can be customers, shops, mobile phone

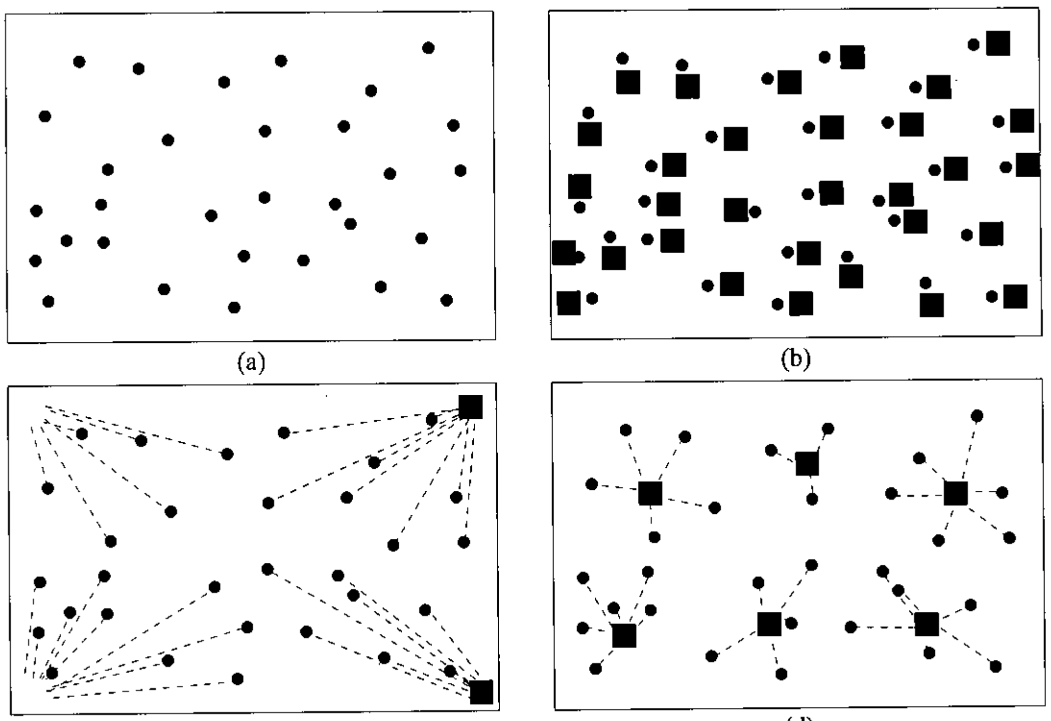


Figure 1.1: The facility location problem

users etc.) who needs to receive a certain service from some facility providers (can be markets, warehouses, mobile phone towers etc.). The goal is to locate the facilities in such a way that a best trade-off is established between the cumulative benefit of the receivers and the cost of setting up and maintaining the facilities and the cost of getting the service (e.g., transportation cost to go to a market). Researchers have proposed different computational model to represent the problem and different ways to find the best solution.

In this thesis, we propose to model the relationship between the service receiver and the service providers through *Voronoi neighborships*. According to the model, a warehouse should be set up in such a location that it gets the maximum number of *Voronoi neighbors* (i.e. can serve as many shops as possible). This way we can ensure that the set up cost is as low as possible and we also ensure that the service cost is low because Voronoi neighbors are geometrically closest by definition. In this thesis, we identify various properties of Voronoi neighbors and use them to devise algorithms to maximize or optimize neighborships.

Now imagine once again that you are the owner of a shopping chain, and you are planning to set up shops in a city where at the moment you do not have any shops, but one of your competitor shopping chain has already set up numerous shops in the city (Figure 1.2(a) shows the locations of the opponents shops using solid circles). You know that your product is at least as good as the opponent's and you also know that if you set up a shop, many customers would be tempted to come to your shop rather than your opponent's. Your objective is to entice customers away from as many shops of your opponent as possible.

It is common to assume that customers prefer to go to the shop which is geographically closest to them (provided that the service is similar in quality), and thus, if we draw the Voronoi diagram, then the Voronoi region corresponding to a shop will determine the customers affiliation with a shop (see Figure 1.2(b)). So, if a new shop is placed it will take away some portion from the Voronoi regions (in other words, customers) of each of its neighboring shops. For example, Figure 1.2(c) shows a new shop (represented as a rectangle) which takes customers away from three existing shops and Figure

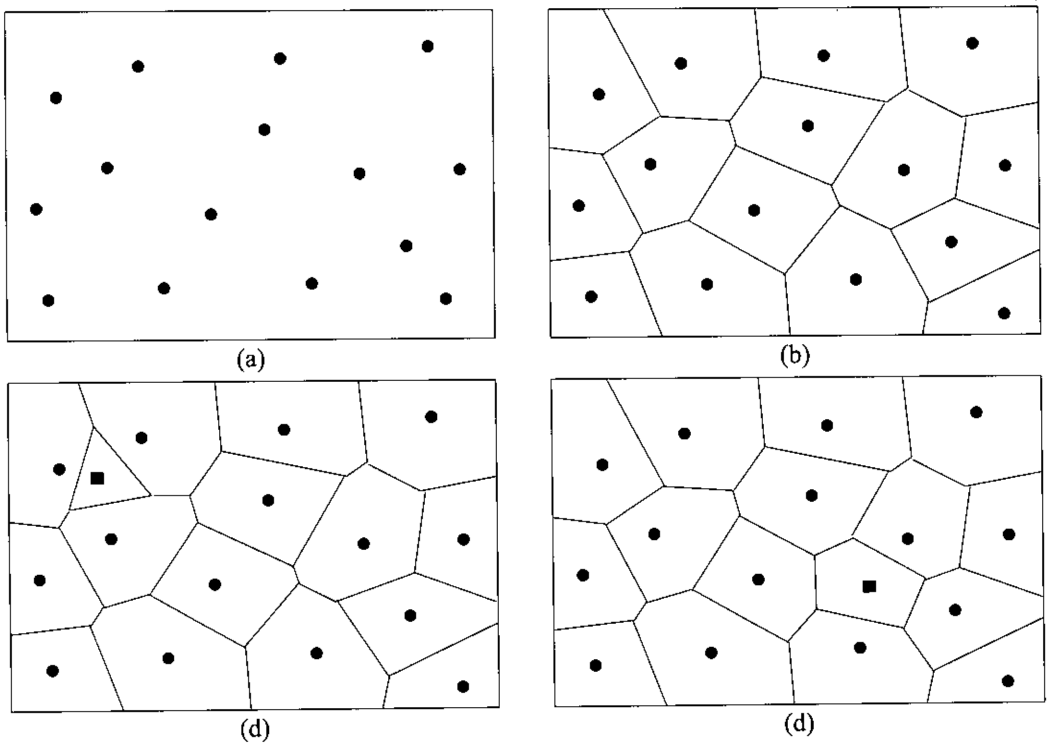


Figure 1.2: Games in facility location

1.2(d) shows another placement where the new shop takes customers away from five existing shops. Therefore, to cause the maximum interference to the business of your opponent you should choose to place your shops such that each of your shops gets as many shops of the opponent as neighbors as possible. With this idea in focus, we introduce several variations of games based on Voronoi neighborships.

The following section gives a brief outline of the problems addressed by this thesis and their results.

## 1.1 The problems and the results

The thesis addresses two problems related to the maximization of the number of Voronoi neighbors. The first problem is to find the location where a new Voronoi site can be placed so that it gets the maximum number of neighbors. This problem is analogous to finding the location where a new warehouse can be set up so that it can serve the maximum number of shops cheaply. The second problem is to find the locations where new *Voronoi sites* can be placed to ensure that a minimum number of new sites is used to get all the existing Voronoi sites as neighbors. This problem is analogous to finding the locations where warehouses can be placed to ensure that all the shops can be served by setting up as few warehouses as possible.

To solve the problems, we first establish that the number of neighbors of a new site depends on the *arrangement of Delaunay circles* generated by the existing Voronoi sites. We also discuss special properties of such arrangements. Based on the properties we construct the *dual graph of the arrangement* and discuss its properties. We prove that to solve the first problem, the new site must be placed in a *cell of the arrangement* corresponding to a *top level node* of the *dual graph.* We also prove that the second problem reduces to the *set cover problem*, where the sets consists of the set of neighbors ensured by the nodes of the dual graph which has no *parents.*

Then we introduce games where two players places Voronoi sites in a plane and compete each other to maximize the number of Voronoi neighbors of the sites placed by them. We devise multiple variation of the game depending on the number of rounds and the specific criterion for winning a game. For some of these games we establish the *winning strategies*, for some games we show that winning is impossible and for some games we show that winning is possible, but only if one of the players makes a non-optimal move.

We also introduce a concept of *Voronoi layers* and develop an algorithm to construct the arrangement of Delaunay circles which can detect all possible intersections of *Delaunay circles* which are within two Voronoi layers.

## 1.2 Outline of the thesis

This section gives a glimpse of the next chapters of the thesis.

Chapter 2 discusses a few basic topics which are very important and relevant for the problems and are often referred to in later chapters or used to discuss or prove the results in the thesis. Chapter 3 discusses previous researches which address problems similar to that discussed in this thesis. Chapter 4 discusses Voronoi neighborships in detail and presents algorithms for the optimization of Voronoi neighbors. Chapter 5 discusses a number of games based on the idea of maximizing the number of Voronoi neighbors and devises winning strategies for them. Chapter 6 discusses a few supplementary ideas and results which are relevant to but not strictly related to the main problems of the thesis. Finally, Chapter 7 concludes the thesis with a summary and also opens up a few avenues of further research.

# Chapter 2

# Preliminary

In this chapter, we are going to explore some preliminary topics that are related to the thesis. These topics include a brief overview of Voronoi diagrams and Delaunay triangulations and their properties, many of which are very important and integral to the results of the thesis. We also give a sketch of only a few of many applications of Voronoi diagrams to emphasize the importance of research in this area. We gave a summary of various kinds of facility location problems which would enable the reader to get a better understanding of the motivation of this thesis. And finally, we discuss some basic topics related to arrangement of geometric objects and combinatorial games, both of which will be referred multiple times in the following chapters.

## 2.1 Voronoi diagram and Delaunay tessellation

Voronoi diagram and Delaunay tessellations, though pretty well known in computational geometry, deserve a section in this discussion because they form the core of the topic of this thesis.

## 2.1.1 Voronoi diagram and its properties

A *Voronoi diagram*, named after Georgy Voronoi, also called a *Voronoi tessellation*, a *Voronoi decomposition*, or a *Dirichlet tessellation* (after Lejeune Dirichlet) [11], is a special kind of decomposition of a metric space determined by distances to a specified discrete set of objects or sites in the space. Commonly, we are given a set of points $S$, and the Voronoi diagram for $S$ is the partition of the plane into regions each of which are associated with one point in $S$. In the rest of the thesis, we shall refer to the points in S as *Voronoi sites* to avoid ambiguity with the general usage of the term 'point' which will be used to indicate any general point in the plane.

The region associated with a site $p$, called *Voronoi region* of $p$ and denoted as $V(p)$, is formed in such a way that all points in $V(p)$ are closer to $p$ than to any other site in $S$. Intuitively, if we start expanding a circle centered at any point $x$ in the plane, then $x$ belongs to the Voronoi region of the first site reached by the circle. Another intuitive way to understand Voronoi regions is - if we imagine $n$ circles expanding from the sites at the same speed, the fate of each point $x$ of the plane is determined by those sites whose circles reach $x$ first. In 2D, the Voronoi regions are always *convex polygons.*

The shared edge between two adjacent Voronoi regions consists of points which are equidistant from the two sites associated with those regions; in other words, it is the perpendicular bisector of the two sites. Such edges are called *Voronoi edges.* The average number of edges of a Voronoi region is less than or equal to six [6].

An intersection of Voronoi edges is called a *Voronoi vertex*. A Voronoi vertex can be formed by the intersection of many edges. But if the sites are in *general positions* (no four sites are co-circular and no three sites are collinear) then all the Voronoi vertices has exactly three edges incident on them. The average number of vertices of a Voronoi region is less than or equal to six.

Two sites are called *Voronoi neighbors* of each other if they share a Voronoi edge; i.e. if their Voronoi regions are adjacent.

Figure 2.1 provides example for the terms described in this subsection. In the figure, the Voronoi sites are shown as solid black circles and are named $p, ..., x$. The Voronoi diagram is drawn using solid lines. The Voronoi vertices are named $1, ..., 10$. Line segments joining the vertices are the Voronoi edges. The Voronoi region of $p$ is shown in gray and has $1, 2, 3, 4$ and 5 on its boundary.

### 2.1.2 Delaunay tessellation and its properties

*Delaunay tessellation* is the *straight-line dual* of the Voronoi diagram. Each node of the Delaunay tessellation corresponds to a Voronoi region and is usually drawn exactly in the same location of the sites and each edge in the tessellation corresponds to a Voronoi edge. The tessellation is a triangulation if no four sites are co-circular. An interesting property of the *Delaunay triangulation* is that, compared to any other triangulation, it maximizes the minimum angle; but it does not necessarily minimize the maximum angle [11].

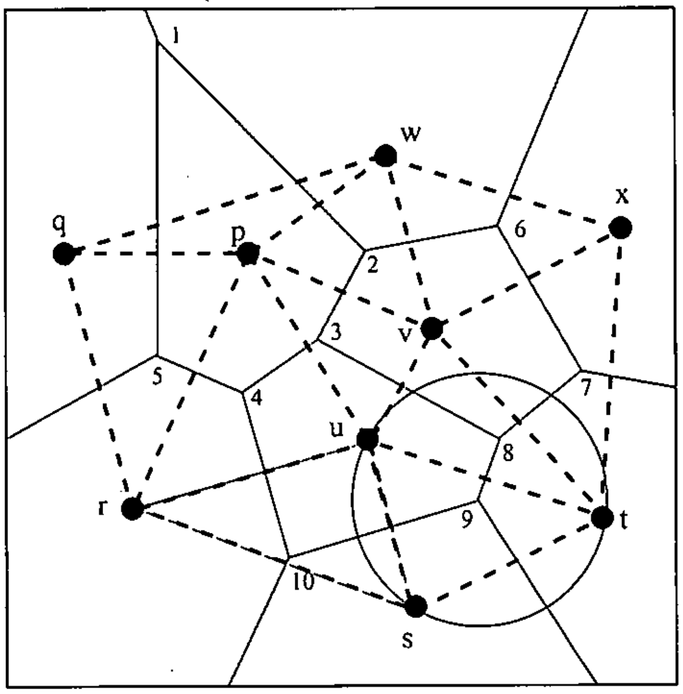


Figure 2.1: Voronoi diagram and Delaunay triangulation

Every triangle of the triangulation is called a *Delaunay triangle*. The union of the Delaunay triangles forms the *convex hull*. The Voronoi sites on the convex hull have unbounded Voronoi regions.

The circum-circle of a Delaunay triangle is called a *Delaunay circle*. Interesting properties of Delaunay circles are - a Delaunay circle is centered at the Voronoi vertex common to the Voronoi regions of the three sites that formed the corresponding Delaunay triangle and a Delaunay circle does not contain any other site inside it. The number of Voronoi vertices (and thus Delaunay circles) for $n$ Voronoi sites is less than $2n-5$ where $n$ is the number of Voronoi sites [11].

Both Voronoi diagram and Delaunay triangulation can be generated in

$O(nlgn)$ time using only linear space [6, 11, 15].

Figure 2.1 also provides example for the terms described in this subsection. In the figure, the Delaunay triangulation is drawn using dashed lines. One Delaunay triangle $\Delta rsu$ is shaded and one Delaunay circle centered at 9 and going through $u$, $s$ and $t$ is also shown. The sites on the convex hull are $q$, $r$, $s$, $t$, $x$ and $w$.

## 2.2 Some important applications of Voronoi diagram

Voronoi diagram and Delaunay triangulations have found a lot of applications in several areas like wireless networks, facility location, nearest neighbor searches, motion planning etc. to name a few.

### 2.2.1 Wireless networks

In wireless networks, Voronoi diagrams are used to address many different problems. One widespread application is the formation of clusters and the choice of cluster-heads. Another application is in finding the minimum exposure path through an area infested with sensor nodes.

### 2.2.2 Facility location

Voronoi diagrams are often used to model customer response or affiliation to a service provider and also to model support network between stores and warehouses. We will elaborate this application more on later sections and chapters.

### 2.2.3 Nearest neighbor queries

The basic idea of nearest neighbor queries is given a set of points, finding the nearest points for each of the points in the set. This can easily be solved by constructing the Voronoi diagram and the nearest neighbor of a point must be among the sites of neighboring regions. This core idea is applied in string comparisons, data mining, clustering etc.

### 2.2.4 Motion planning

Given an initial and ending position for an autonomous robot and many obstacles in between, what should be the best path for the robot? The solution is to draw the Voronoi diagram and follow the Voronoi edges. Similar solutions can also be applied in developing AI for computer games.

## 2.3 Facility location problems

The basic question in *facility location problem* is- where should I place a new facility to serve some facility receivers such that the benefits are maximized? Examples of *facility or service providers* can be a warehouse, market, mosque etc. which provides a specific service to its clients. On the other hand *service/facility receivers* corresponding to the providers stated above can be shops, customers, devotees etc. who receive the service.

The exact formulation and solution of facility location problem depends on a lot of factor like the position of the other facilities, the customers or facility receivers, the nature of the affiliation with the receivers, the definition and beneficiary of the benefit etc. Widely we can classify the facility location problem into three closely related variations which are outlined here. Several researches addressing these problems will be discussed in Chapter 3

### 2.3.1 Cooperative facility location

The common example of this type of problem is the one we already mentioned in Chapter 1 where there are multiple shops or retailers and we want to place warehouses in such a way that ensures maximum benefit to both the shops and the warehouse owner. The objective is usually to reduce the total service cost (cost for a shop to transport goods from the warehouse) and the facility cost (cost to setup and maintain the warehouse). It is usually assumed that every shop is committed to get connected to the warehouse which is geometrically closest (or maybe takes least amount of time to reach).

The same model is applicable when we want to place a facility to serve the general population. For example: setting up mosques, community centers, schools etc. We want to place the facility in such a location that it provides maximum benefit to the most number of peoples.

### 2.3.2 Non-cooperative facility location

The non-cooperative model adds another parameter to the cooperative facility location problem. Here the customers or facility receivers have the ability to choose not to take service from a facility. They might decide to use a facility which is not the closest due to various reasons. One such case maybe, that the customers prefer to go to a facility which is further but is not as overloaded as the closer one.

Another quite interesting variation is where the customers are not satisfied with the current locations of the facilities and they might themselves try to establish a new facility for their benefit. In such cases, the facility location problem has to consider the service cost, facility cost and as well as best possible coverage of the population (a good coverage minimizes the maximum service cost).

### 2.3.3 Competitive facility location

In *competitive facility location*, the problem gets another twist in the sense that now two or more service providers competes each other to attain the patronage the maximum number of service receivers. For example, Pizza Hut and Dominos are competitor entities who provide similar services to their clients. So assuming that clients always go to the outlet which is nearest (geographic distance or time required to reach), the owners of the two companies would compete to set up their outlets in such locations that they get more customers than their opponent.

## 2.4 Arrangement of geometric objects

In *computational geometry*, *arrangement* refers to the decomposition of the space into *cells* by a collection of intersecting objects. To generate an arrangement means to find out the exact number of cells that are formed, the *intersection points*, the *boundaries of the cells* and also which objects intersected to form which cells etc. In two dimensions, we can consider arrangement of lines, polygons, circles etc.

For example, if the arrangement consists of three parallel lines, then there are four regions, none of them bounded. If we add a line crossing the three parallels, then there will be eight regions, still none of them bounded. If we add one more line, parallel to the last, then there will be 12 regions, of which two are bounded parallelograms.

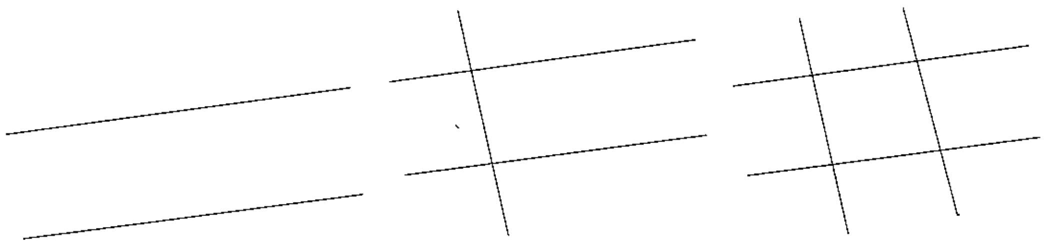


Figure 2.2: Example of arrangement of straight lines

For our thesis, we are only interested in the arrangement of multiple circles in 2D. In an *arrangement of circles*, we need to know exactly how many cells there are, which are the intersection points, which are the arcs that form the boundary of the cells and which circles they belong to. For example the following figure shows the arrangement of three circles. They have formed 5 cells. Cells $A$ and $B$ are part of only one circle; cells $C$ and $D$ are part of two circles and cell $E$ is part of three circles.

Agarwal and Sharir [2] have given a deterministic algorithm to compute the arrangement of $n$ circles in 2D. The time complexity of the algorithm is $O(n\lambda_4(n))$, where $\lambda_4(n)$ is the length of the *Davenport-Schinzel sequence* of order 4 with $n$ symbols. Davenport-Schinzel sequence of order $s$ with $n$

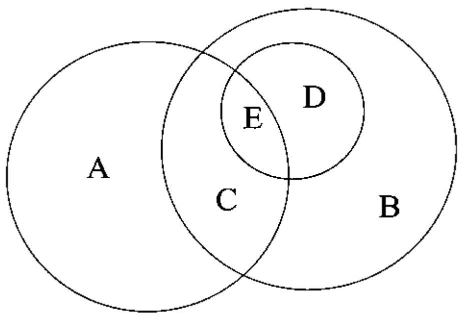


Figure 2.3: Example of arrangement of circles

symbols, $\lambda s(n)$, is a sequence $(u_1, \cdots, u_m)$ composed from $n$ symbols such that the same symbol does not appear consecutively in the sequence and the sequence does not contain any alternating subsequence that uses two symbols and has length $s+2$. The standard bounds for $\lambda_4(n)$ is $\Theta(n.2\alpha(n))$ where $\alpha(n)$ is the *inverse Ackerman function* which is extremely slow growing. In fact, if $n$ is less than or equal to an *exponential tower* of 65536 2s, then $\alpha(n) \leq 4$. In conclusion, we can state that even for very large values of $n$, the arrangement of $n$ circles can be computed in close to $O(n^2)$ time.

For more information on arrangments and their applications interested readers can refer to [3, 1].

## 2.5 Game theory and combinatorial games

*Game theory* is a branch of applied mathematics that is used in the social sciences (most notably economics), biology, political science, computer science, and philosophy. Game theory attempts to mathematically capture behavior in strategic situations, in which an individual's success in making choices depends on the choices of others. Games can be cooperative or non-cooperative, symmetric or asymmetric, simultaneous or sequential, perfect information or imperfect information, discrete or continuous, infinite or defined etc. combinatorial games are a subset of these which are sequential, discrete and have perfect information.

*Combinatorial game theory (CGT)* usually studies two-player games in which the players take turns to change the state of the game in defined ways or moves to achieve a defined *winning condition* [7]. Here the set of possible *states* and the set of possible moves at each state are strictly defined. In fact the progress of the game can be represented as a rooted tree, called *state tree*, with the initial state as the root and the other nodes representing the other possible states of the games, an edge between two nodes represent the possibility of a move that can cause the game to move from one state to the other. The collection of outgoing edges at each node represents the possible choices for the next move when the game is in that state. The game also has a few states which are defined as the winning states and are represented by the leaves of the tree. In some cases, when the game can end in a tie, there can be leaves which do not represent winning states. The objective of the players is to reach a winning leaf before the other player.

Combinatorial game theory does not study *games of chance* (for example, games involving throwing a dice etc.), but restricts itself to games whose position is public to both players, and in which the set of available moves is also public. This attribute is referred to as *perfect information.* CGT principles can be applied to games like chess, checkers etc. But these games are mostly too complicated to allow complete analysis. combinatorial games are called *impartial* if both players have the same set of allowed moves in each position of the game.

Any impartial perfect-information combinatorial game without ties has one of two outcomes under optimal play (when the players do their best to win): a first-player win or a second-player win. In other words, whoever moves first can force himself to reach a winning leaf, or else whoever moves second can force himself to reach a winning leaf, no matter how the other player moves throughout the game. Such forcing procedures are called *winning strategies*.

The target of CGT is to determine the optimum sequence of moves for both players until the game ends, thus discovering the optimum move in any position and to find the winning strategy for either of the players. It is widely accepted (e.g. [7]) that all combinatorial games can be reduced to the classic game called *Nim* and finding a winning strategy for any game is equivalent to finding a *Nim-value* for the game.

For more information abour CGT, readers might refer to the survey [13]. A regularly updated bibliography of researches related to CGT can be found in [16] (contains 1160 entries when this thesis is being written).

The games we address cannot be tagged as strictly combinatorial games. Because though we consider games having two players who play sequentially, there is no uncertainty or randomness, and we have a definite set of possible states of the game (with a bit of abstraction), but the possible set of moves are infinite. So we cannot map the games to Nim and we have to solve them uniquely.

# Chapter 3

# Related Works

In this chapter, we want to mention a few research works which are relevant to our thesis.

First, we will discuss some results which use Voronoi diagrams to model competitive facility location problems and address issues like maximizing the Voronoi area and finding winning strategies for games where the winning criteria involve maximization of Voronoi area.

F. Dehne, R. Klein and R. Seidel [12] worked with the goal of finding the location where a new Voronoi site can be placed so that the area of its Voronoi region is maximized. They formulated the area of the new site as a mathematical function with the location of the new site in cartesian coordinates as the variable and the location of the neighbors as constants.

Here, they assumed that the location of the neighbors are convex, i.e. if they are connected sequentially, then they will form a convex polygon. Then, they evaluate the function along the loci where the new site can be placed such that those existing sites are its neighbors. They proved that if the neighbors are in convex positions then there can be exactly one maxima

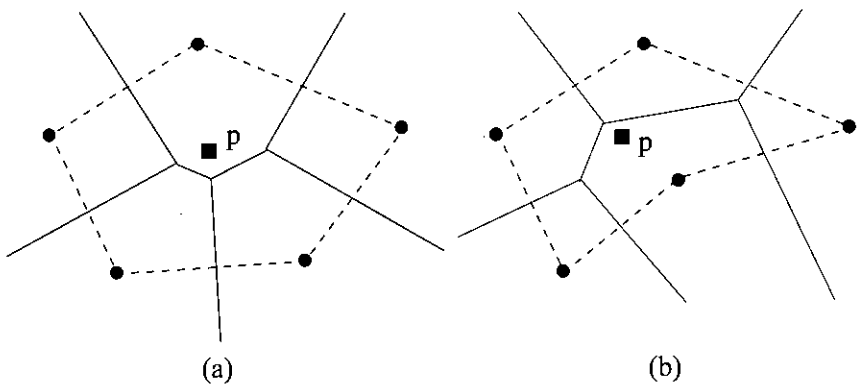


Figure 3.1: Maximizing Voronoi region

of the function and that is where the new site should be placed to ensure that the area is maximized. The limitations of this work is that it does not work for the cases where the neighbors are in non-convex positions. As the technique is not applicable for all situations, it cannot be extended to work globally, i.e. for the entire Voronoi diagram. For example, in figure 3.1 (a), the neighboring sites of $p$ are in convex positions and this algorithm can be applied to identify the location where $p$ should be placed such that the area of $V(p)$ is maximized. But in Figure 3.1 (b), the neighboring sites are in concave positions and this algorithm might fail.

O. Cheong, A. Efrat, S. Har-Peled also worked with a similar goal in [9]. They claimed that though the area of the Voronoi region of a new site can be represented in a closed formula, the handling and comparison for specific input values might become cumbersome and it is unknown whether it is $NP$ or not. So, they tried to find approximate solutions to the problem. They showed that given a set $T$ of $n$ points and a $\delta > 0$, it is possible to find a point $x_{app}$ such that $\mu(x_{app}) \geq (1-\delta)\mu_{opt}$, where $\mu(x)$ is the area of the Voronoi region of $x$ in the Voronoi diagram of $T \cup x$, and $\mu_{opt} = max_x \mu(x)$. the running time of their algorithm is $O(n/\delta^2 + n \log n)$.

O. Cheong, S. Har-Peled, N. Linial and J. Matoušek [10] addressed one round Voronoi game. According to their formulation, two players place their sites inside a unit square region and try to maximize the total Voronoi area of the Voronoi regions of their sites. Due to the game being one-round, the first player places all $n$ of his sites first and then the second player places his $n$ sites. They proved that, for large enough $n$, the second player can always place his sites in such a way that the sum of the areas of their Voronoi regions is at least $1/2 + \alpha$ portion of the total playing area, where $\alpha > 0$ is a constant independant of $n$. In other words, the found a winning strategy for the second player.

S. P. Fekete and H. Meijer [14] extended the results of Cheong et al to find winning strategies even when the playing area is not an unit square. They used a parameter $\rho$ to represent the aspect ratio of the playing area. And they showed that the second player wins only if $n \geq 3$ and $\rho > \sqrt{2}/n$ or if $n = 2$ and $\rho > \sqrt{3}/2$. In all other cases, the first player has a winning strategy. They also proved that if the playing area is a polygon with holes, then finding a winning strategy for the second player is *NP-hard.*

H.-K Ahn, S.-W Cheong, O. Cheong, M. Golin and R. van Oostrum [4, 5] addressed n-round Voronoi games. Here, the players take turns to place their sites. They are only allowed to place one site in each round and in total they place $n$ sites in $n$ rounds. They could give winning strategy for the second player only if the playing area is one dimentional, i.e. a line segment or a circle. For two dimentional playing area the problem is still open.

We must mention that, besides Voronoi diagrams, many different models and techniques have been used to model and solve different variations of facility location problems. For example, T. Moscibroda and R. Wattenhofer [22] modeled the cooperative facility location problem using bipartite graphs where the edges are weighted using service costs and they give a distributed approximation algorithm to optimize the overall benefit. D. B. Shmoys, E. Tardos and K. Aardal [23, 24] also addressed similar problems and gave approximation algorithms. On the other hand, rather than highlighting the total cost, M. X. Goemans and M. Skutella [17] highlighted the distribution of the cost to the customers. They argued that if the cost is not fairly allocated, then the customers might choose to open their own facilty or ask a competitor company to serve them. The authors used linear programming relaxation to identify if fair cost allocation is possible or not. Interested readers may also like to go through [8, 19] for some other related results.

# Chapter 4

# Optimizing Voronoi neighbors

In this chapter, we will discuss two problems related to the optimization or maximization of Voronoi neighbors, with applications in facility location.

The first problem is: given $n$ Voronoi sites in the plane, to find the location where a new Voronoi site can be placed so that it gets the maximum possible number of Voronoi neighbors. For example, in Figure 4.1, it is shown how the placement of the new site plays a role in the number of neighbors it is going to get. Figure 4.1(a) shows the initial sites and the Figures 4.1(b), (c) and (d) shows three different placement options of a new site $p$ and we can see the location where it gets the maximum number of neighbors in Figure 4.1(d). Our objective is to find a deterministic technique to find such location(s) in a Voronoi diagram.

The second problem is: given $n$ Voronoi sites in the plane, to find the minimum number of new sites that has to be placed (as well as the location where they should be placed), so that all the existing $n$ sites become neighbors to at least one of the new sites. For example, Figure 4.2(a) shows that ten sites were given and the next figures shows how new sites are added to get

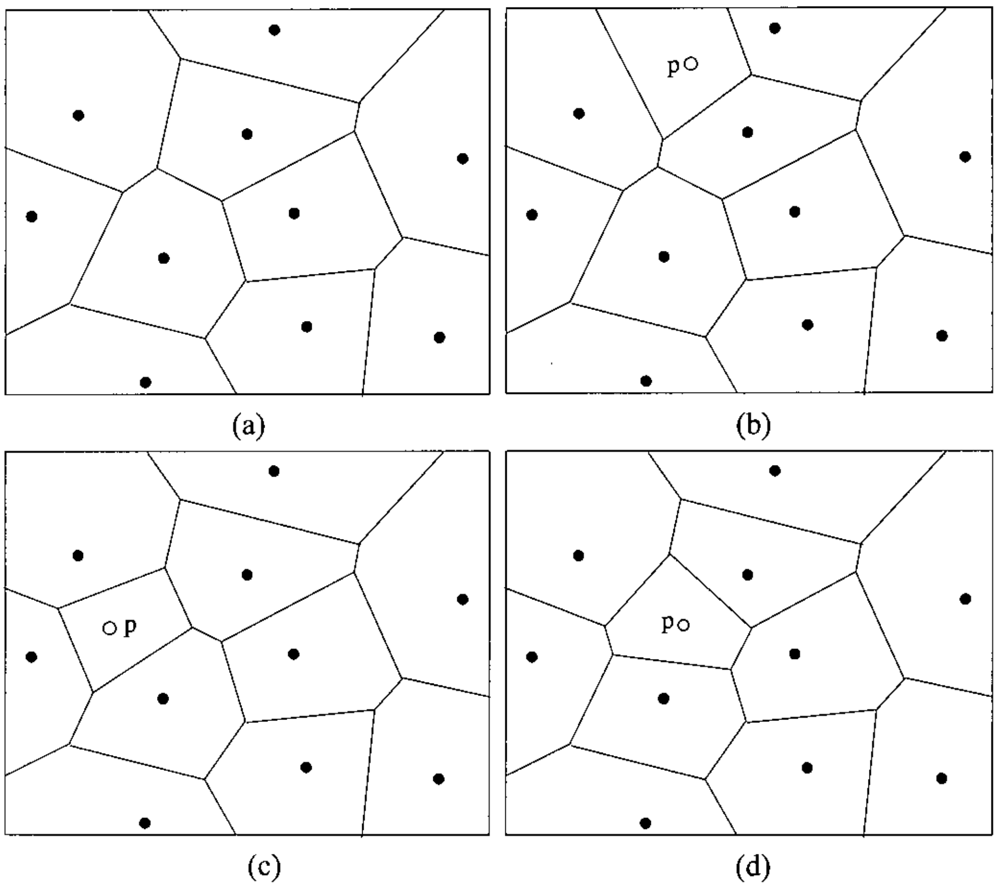


Figure 4.1: Elaboration of the first problem

all of the ten points as neighbors. The first placement (Figure 4.2(b)) uses four new sites and the second placement (Figure 4.2(c)) uses only three new sites which is the minimum number of new sites needed for this example.

The next sections deal with these two problems. But, before that we discuss the basics of Voronoi neighborship and the arrangement of Delaunay circles and their properties, we discuss how these properties enable us to construct a Dual graph of the arrangement and finally, we discuss how the maximization and optimization problems can be solved with the help of the

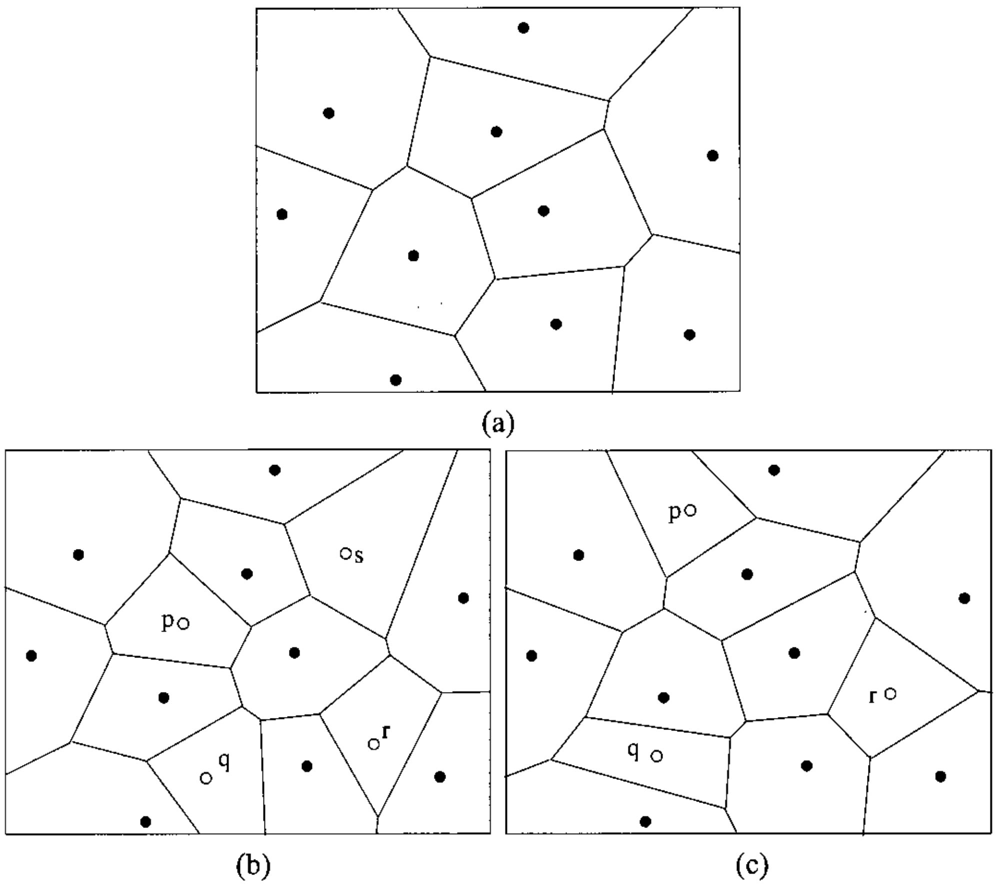


Figure 4.2: Elaboration of the second problem

dual graph.

## 4.1 Voronoi neighbors and arrangement of Delaunay circles

This section is devoted to the discussion of Voronoi neighborship, their properties, how they are related to the arrangement of Delaunay circles and the properties of the arrangement.

### 4.1.1 Becoming neighbors

In this subsection, we are going to discuss the rules of becoming or aquiring Voronoi neighbors. More precisely, we want to answer the question - when a new site is placed, which of the existing sites become its neighbors and why? The following lemma gives the answer.

**Lemma 4.1.1.** *If a new site, p is placed inside an existing Delaunay circle, then the three existing sites a, b and c which are on that circle becomes its neighbors. If p is placed inside the Delaunay triangle $\Delta abc$, then the triangle remains intact but if p is placed outside the Delaunay triangle $\Delta abc$, then the Delaunay edge, on whose side p is placed, is removed.*

*Proof.* We know that the center of a Delaunay circle is a Voronoi vertex (refer to Section 2.1.1). Let, the center of the Delaunay circle going through $a$, $b$ and $c$ be $v$ (see Figure 4.3(a)). Now if $p$ is placed inside the circle but outside the Delaunay triangle in $ac$'s side (i.e. in the region bounded by the arc $ac$ and the the edge/chord $ac$), then $vp$ will be smaller than $va$, $vb$ and $vc$ (see Figure 4.3(b)). So the point $v$ should be inside the Voronoi region of $p$. And if we move along $vb$, we will find a point, $m$ such that $mb = mp$, $mp < ma$ and $mp < mc$ (see Figure 4.3(c)). So, $b$ will become neighbor to $p$. Similarly, $a$ and $c$ also become neighbors of $p$.

In Figure 4.3(d), the dotted line represents the perpendicular bisector of $ac$, so each point on this line is equidistant from $a$ and $c$. Thus, if any portion of this line is further from other sites than it is from either $a$ or $c$, then $a$ and

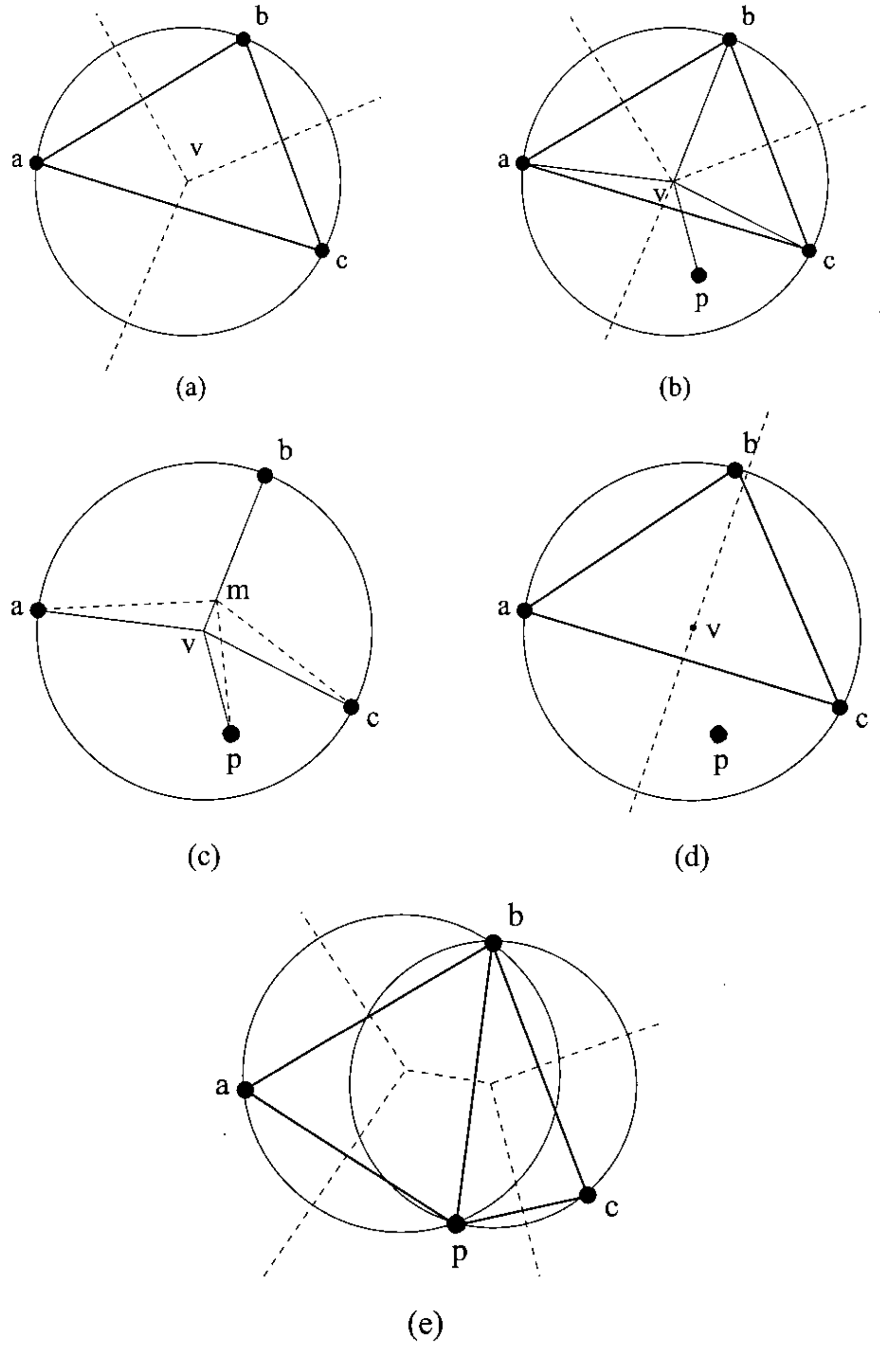


Figure 4.3: When a new site, $p$ is placed inside the Delaunay circle but outside the Delaunay triangle

$c$ remains Voronoi neighbors and the Delaunay edge $ac$ remains. But we can see that any point on this line which is in the portion above $v$, is closer to $b$ than it is to either $a$ or $c$. And any point below and including $v$ is closer to $p$ than it is to $c$ (we assume that $p$ is placed on $c$'s side of the bisector). Therefore, $a$ and $c$ no longer remains neighbors.

The Voronoi diagram is redrawn in the next figure where we see that $p$ becomes neighbors to $a$, $b$ and $c$ (see Figure 4.3(e)) and $a$ and $c$ are no longer neighbors of each other.

Similarly, if $p$ is placed inside the circle and inside the Delaunay triangle(see Figure 4.4(b)), then again $vp$ will be smaller that $va$, $vb$ or $vc$. And $p$ will be neighbors of $a$, $b$ and $c$. But now we shall find a point $m$ on the bisector of $ac$ which is equidistant from $a$, $c$ and $p$ (see Figure 4.4(c)); so, the portion of the bisector below $m$ is closer to $a$ and $c$ than any other sites. So, $a$ and $c$ remains Voronoi neighbors and the Delaunay edge $ac$ remains intact. For the same reason $ab$ and $bc$ also remains intact. Figure 4.4(d) shows the resultant Voronoi diagram and Delaunay triangulation. $\square$

So, if the new site is placed inside the Delaunay triangle, then all of the existing Delaunay edges are intact. But if the site is placed outside the triangle but inside the circle, then one of the Delaunay edges is deleted. This rule is called *edge flipping* and it was introduced in [21] and was used in [18] to compute the Delaunay triangulation. *Edge flipping* means after placing a new site inside a Delaunay circle, we can redraw the triangulation by connecting the new site to the existing sites and deleting any previous

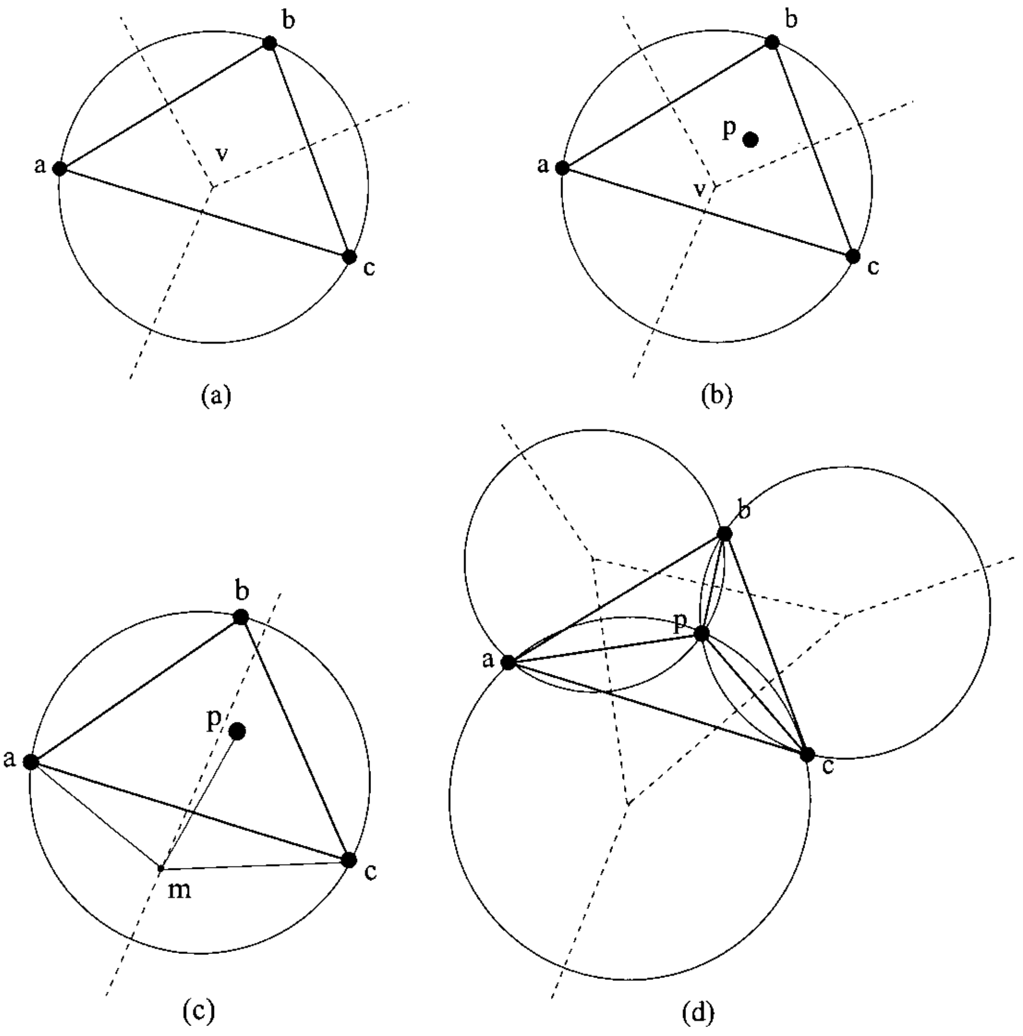


Figure 4.4: When a new Voronoi site, $p$ is placed inside the Delaunay circle and inside the Delaunay triangle

Delaunay edge which intersects any of the newly drawn edges.

Now, if there are more than three Voronoi sites, then there must be more than one Delaunay circles and they will intersect. In such cases, if we place a new site, it might be inside more than one circle (i.e. inside their intersection). The following lemma extends Lemma 4.1.1 for such cases.

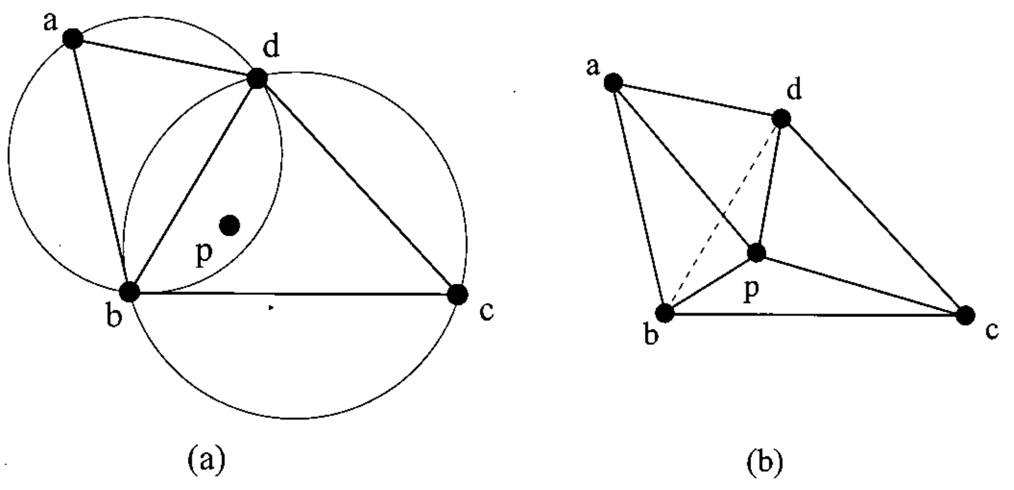


Figure 4.5: When a new Voronoi site, $p$ is placed inside the intersection of more than one Delaunay circles

**Lemma 4.1.2.** *If* $\boldsymbol{C}$ *is the set of all existing Delaunay circles which contain the new site, p inside them, then the set of neighbors of p,* $\boldsymbol{N}(p) = \{x | x$ *is on a Delaunay circle, c where* $c \in \boldsymbol{C}$ $\}$ *and in the new triangulation, p will be connected to all the sites in* $\boldsymbol{N}(p)$ *and any existing Delaunay edges intersecting these new edges will be deleted.*

*Proof.* This result is achieved by applying the rules of Lemma 4.1.1 for each circle $c \in \mathbf{C}$ and taking the union. □

Figure 4.5 gives an example for the rule described in Lemma 4.1.2.

## 4.1.2 Arrangement of Delaunay circles

The discussion in the above section implies that we can find out which of the existing sites become neighbors of the new site if we know exactly which Delaunay circles contain the new site, $p$.

From our discussion of arrangement of circles (see Section 2.4), we know that the arrangement decomposes the plane into cells formed by the intersections of the circles. As circles are convex objects, they can only intersect each other twice and that is why each cell of the arrangement is unique and is formed by the intersection of a unique set of circles.

After computing the arrangement of Delaunay circles, we shall know the exact points of intersections, the boundaries and most importantly the set of circles that intersected to form a particular cell. In other words, if we know the arrangement of Delaunay circles, we can easily find out the cell that contains the newly added site, $p$. And the set of circles that intersected to form that particular cell is precisely the set of all Delaunay circles, $\mathbf{C}$ which contain the new site, $p$ inside them.

**Lemma 4.1.3.** *If a new site $p$ is placed inside a cell $A$ of the arrangement of Delaunay circles, then all the existing sites that are on any of the circles that intersected to form $A$ will become neighbors of $p$.*

*Proof.* The proof follows from the definition of a cell of the arrangement of Delaunay circles and Lemma 4.1.2. $\square$

Now, we want to mention a few natations and assumptions which will be used in the rest of the thesis.

- We will use capital letters (e.g. $A$, $B$ etc.) to name the cells of the arrangement.

- Whenever we mention that a site $p$ is placed inside a cell $A$, it will indicate that it is placed inside the boundary of $A$ and not on any of the arc bounding it so that the sites remain in general positions.
- The set of neighbors that a new site will receive if it is placed inside a particular cell, $A$ will be denoted as $\mathbf{N}(A)$ and will be referred to as *set of neighbors ensured by $A$*.

### 4.1.3 Property of the arrangement of Delaunay circles

In this subsection, we shall explore some properties of the arrangement of Delaunay circles. But before that we want to define *completeness of the arrangement* and *arc-adjacent cells of the arrangement.*

An arrangement of Delaunay circles is *complete* only if the corresponding Delaunay triangulation is complete. A Delaunay triangulation is *complete* when every face except the face outside the convex hull of the triangulation has exactly three edges and no new Delaunay edges can be added without intersecting any existing edges.

Figure 4.6 shows example of complete and incomplete triangulations and the corresponding complete and incomplete arrangement. In Figure 4.6(a) the triangulation is incomplete, because we can add another Delaunay edge to get the complete triangulation shown in Figure 4.6(b) where no new edges can be added. Figure 4.6(c) is the arrangement corresponding to the incomplete triangulation shown in Figure 4.6(a) and Figure 4.6(d) is the complete arrangement corresponding to the complete triangulation shown in Figure

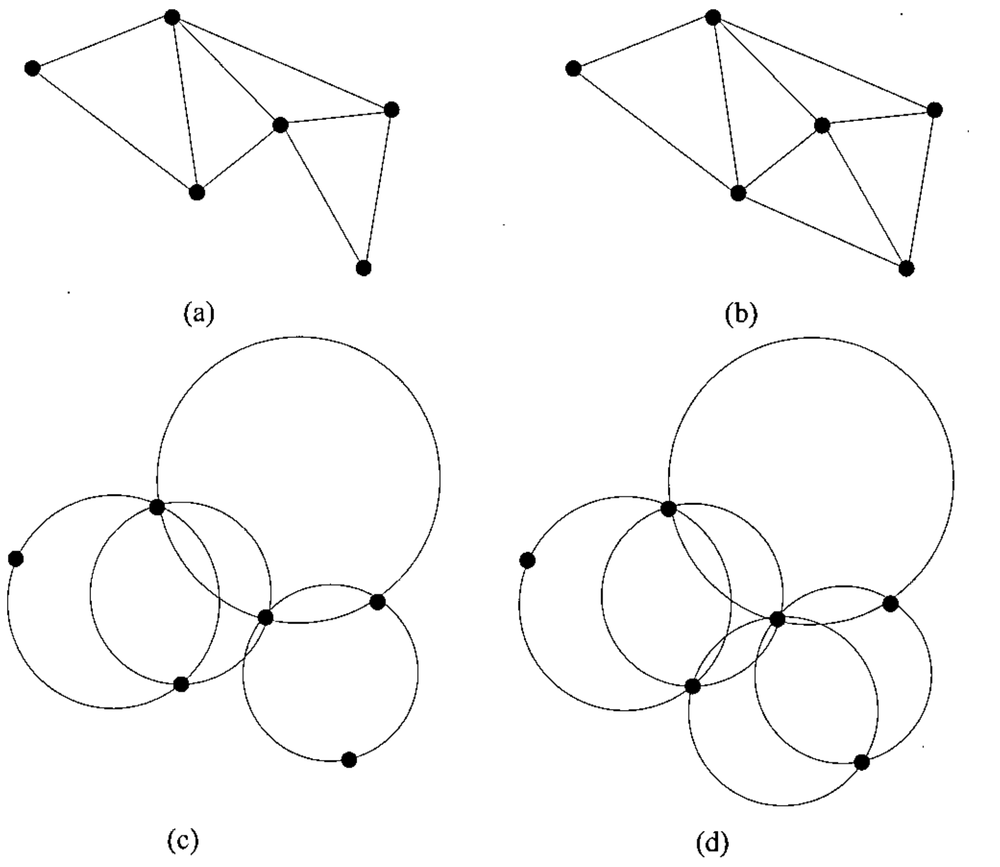


Figure 4.6: Examples of complete and incomplete Delaunay triangulation and their corresponding arrangements

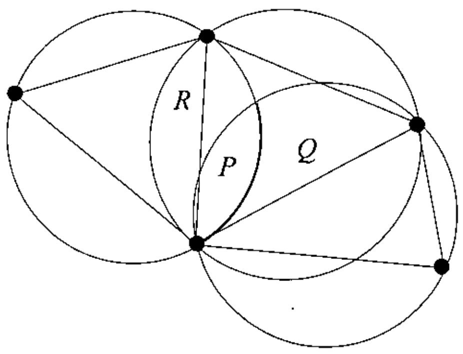


Figure 4.7: Examples of arc-adjacent and non-arc-adjacent cells

4.6(b).

Two cells in the arrangement of Delaunay circles are *arc-adjacent* if they share a common arc. In Figure 4.7 the cells $P$ and $Q$ are arc-adjacent, but cells $Q$ and $R$ are not arc-adjacent.

The following lemmas and theorem present a few properties of the arrangement of Delaunay circles.

**Lemma 4.1.4.** *To add a new circle in a complete arrangement, a new site must be added.*

*Proof.* The proof is given by contradiction.
Let, we can add a new circle without adding new sites. This implies that no new sites are needed and that the new circle goes through three existing sites. But in that case those sites must be on a Delaunay triangle. Therefore, either the triangle and thus the circle is not new, or the triangulation was not complete before, which is a contradiction. □

**Lemma 4.1.5.** *In a* complete arrangement *with at least 4 points, a circle cannot have two Voronoi sites on it which are not on any other circle.*

*Proof.* The Voronoi region of any site which is not on the convex hull must be bounded by at least three Voronoi edges and thus must have at least three neighbors. So, in the Delaunay triangulation, the site must be part of at least three triangles. Thus, it will be on at least three Delaunay circles. Therefore, we can conclude that any site which is not on the convex hull cannot be part of only one circle.

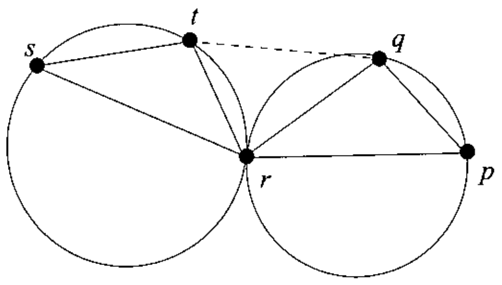


Figure 4.8: A Delaunay circle cannot have two Voronoi sites which are not on other Delaunay circles

On the other hand, some sites on the convex hull can be part of only one triangle and thus on only one circle. So we need to prove that two such points can not be on a single circle. The proof is by contradiction.

Let, $s$, $r$ and $p$ be three consecutive sites on the hull. Let us assume that there is a circle which have two Voronoi sites $p$, $q$ and $r$ on its perimeter and $p$ and $q$ are not on the perimeter of any other circle. Then, it implies that the edge $pq$ belongs to only the triangle $\Delta pqr$ and $p$ and $q$ do not belong to any other triangle. Let $r$ also belongs to another triangle $\Delta rst$. But then the triangulation is not complete because the hull is not convex and we can add at least one(shown in Figure 4.8) more edge to make it complete. □

**Lemma 4.1.6.** *In a* complete arrangement *with at least 4 points, a circle cannot have three Voronoi sites on it which are not on any other circle.*

*Proof.* Follows from Lemma 4.1.4. □

**Theorem 4.1.1.** *Let, $P$ and $Q$ are two arc-adjacent cells in the arrangement of Delaunay circles and the common arc between them is convex with respect*

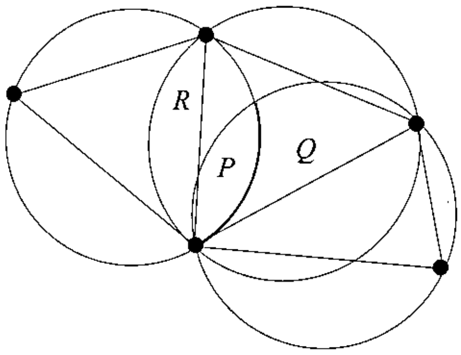


Figure 4.9: The common arc between $P$ and $Q$ is convex with respect to $P$.

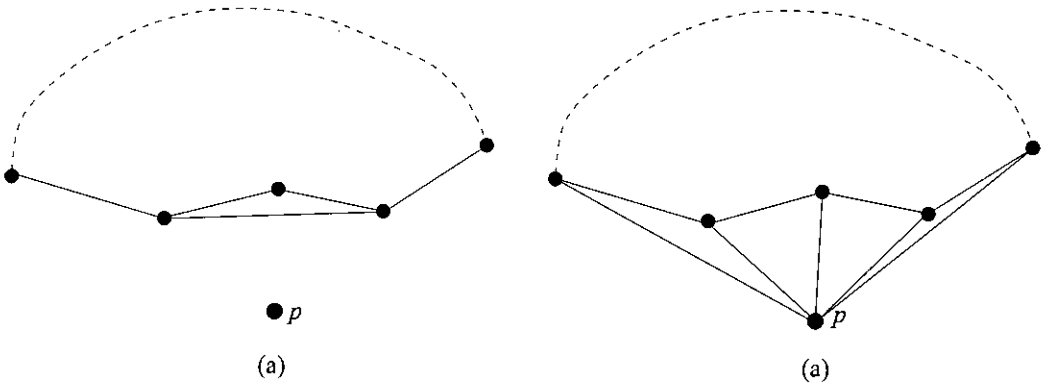


Figure 4.10: a new site $p$ is added outside the hull and the hull is updated

*to* $P$. *Now, if* $\boldsymbol{N}(P)$ *and* $\boldsymbol{N}(Q)$ *are the set of neighbors ensured by them, then* $\boldsymbol{N}(Q) \subset \boldsymbol{N}(P)$ *and* $\boldsymbol{N}(P)$ *has exactly one more element than* $\boldsymbol{N}(Q)$.

*Proof.* Before going into the details of the proof, we will first elaborate the statement of the theorem using an example. Figure 4.9 shows an arrangement of three circles and the thick arc is the common arc between two arc-adjacent cells $P$ and $Q$ and the arc is convex with respect to $P$. The arc being convex with respect to $P$, indicates that $P$ is part of the circle corresponding to that arc and $Q$ is not part of that circle.

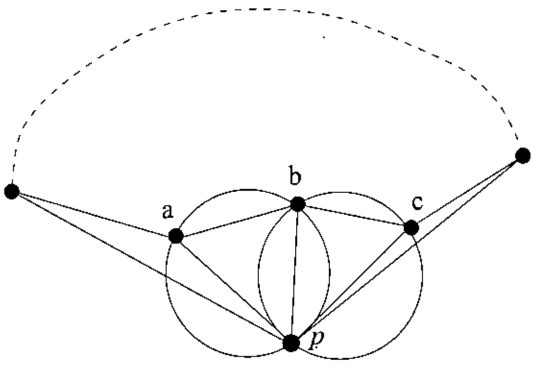


Figure 4.11: When a new circle intersects another new circle

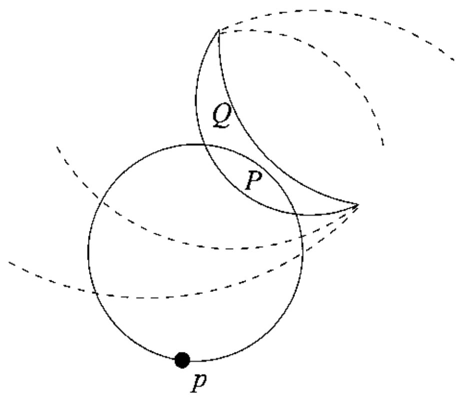


Figure 4.12: When a new circle intersects an existing cell

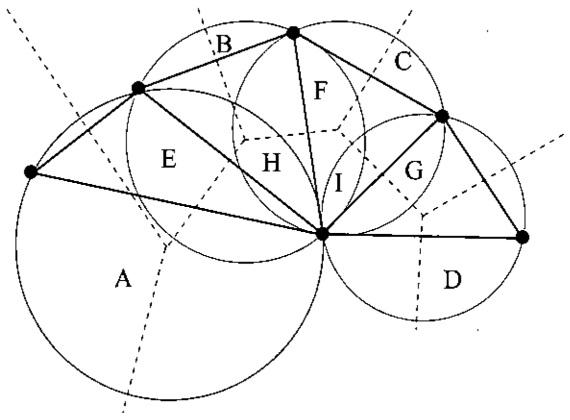
Now, we will prove it using a constructive technique of generating the Delaunay triangulation. In this technique the triangulation is generated iteratively starting with three sites and their convex hull. Then, another site is added outside the convex hull and the hull is updated using the rules of edge-flipping [18]. This way the entire diagram is built. Figure 4.10 elaborates the idea of updating the hull. We see that some new triangles are formed (and some previous ones might be destroyed) and each of the new triangles consists of two existing sites and the new site $p$.

Now, when a new circle intersects another new circle, for example- if $a$, $b$ and $c$ are three consecutive existing sites and $p$ is the new site then $\Delta abp$ and $\Delta bcp$ will be two Delaunay triangle and their corresponding Denaulay circles will intersect (see Figure 4.11. In that case, the cell formed by the intersection will ensure exactly one more neighbor than the two arc-adjacent cells (remaining portion of the circles). Similar arguments can be used if a third new Delaunay circle $\Delta cdp$ intersects them and even if more new Delaunay circles get involved.

Again, if a new circle, $C_p$ intersects any existing cell, $Q$ in the arrangement (see Figure 4.12) then the cell will be divided into two parts and the newly formed part (let's call it $P$), which is inside $C_p$ will ensure exactly one extra neighbor, namely $p$; the two parts are otherwise equivalent. Thus $P$ ensures exactly one extra neighbor. $\square$

## 4.2 Dual graph of the arrangement

From Section 4.1, it is clear that the complete knowledge of the arrangement of Delaunay circles is necessary to know the locations where new Voronoi sites can be placed to maximize/minimize its neighbors. In this section we present the idea of constructing the dual graph of the arrangement of Delaunay circles to better model and utilize the hierarchical relationship existent among the cells of the arrangement.



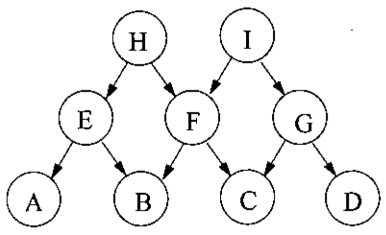


Figure 4.13: Example of dual graph of the arrangement of Delaunay circles

### 4.2.1 Construction of the dual graph

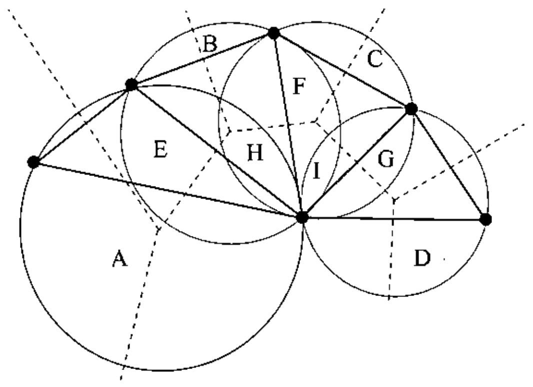
According to Theorem 4.1.1, we know that for any two arc-adjacent cells, the number of neighbors ensured by the cells varies by exactly one. Using this property we can define the dual graph, $G$ as-

- the set of vertices, $\mathbf{V}(G) = \{X \mid X \text{ is a cell of the arrangement}\}$ and
- the set of directed edges, $\mathbf{E}(G) = \{XY \mid X \text{ and } Y \text{ are arc-adjacent cells of the arrangement and } \mathbf{N}(Y) \subset \mathbf{N}(X)\}$.

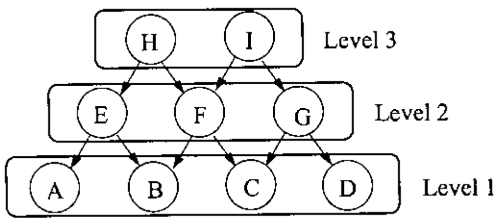
Figure 4.13 gives an example of one such dual graph.

### 4.2.2 Property of the dual graph

A property of the dual graph follows from Theorem 4.1.1 that all the *leaf nodes* correspond to cells of the arrangement which are part of just one circle and thus ensure exactly three neighbors. Then, the nodes which are parents



Arrangement of Delaunay Circles



Levels in the Dual Graph of the Arrangement of Delaunay Circles

Figure 4.14: Levels in the dual graph of the arrangement of Delaunay circles

of any node any leaf node are part of exactly two circles and ensure exactly four neighbors. To better represent this property, we define *levels* in this dual graph as follows-

- *Set of nodes in Level-1*, $L_1 = \{X \mid X$ is a *leaf* in the Dual graph of the arrangement$\}$ and
- *Set of nodes in Level-n*, $L_n = \{X \mid \exists_Y [Y \in L_{n-1}$ and $X$ is a *parent node* of $Y$ in the dual graph$]\}$.

Figure 4.14 provides an example of levels in the dual graph.

According to Theorem 4.1.1, the difference between the number of neighbors ensured by two arc-adjacent cells is exactly one. So, all the children of a node ensure the same number of neighbors and thus are in the same level and though the level of a node is defined recursively, it will always be unique. And another aspect worth noting is that, because a node can not be a parent of any node which are in a higher level or even in the same level, there can be no cycles in this dual graph. Also note that the nodes that have no parents are not restricted to the top level only.

### 4.2.3 Data structure

For maintaining the dual graph, for each node we propose to store its name, its level, the list of its children and whether it has any parent or not. We also propose to maintain a list of nodes in the top level and a list of nodes having no parents to speed up the solution of the problems we are addressing.

### 4.2.4 Time complexity

The dual graph can be constructed while the arrangement is being computed by creating a node whenever a new cell is identified. The existing algorithms (see section 2.4 for reference) computes the arrangement in $O(n\lambda_4(n))$ time and the time complexity for constructing the dual graph is the same.

## 4.3 Maximization and its variation

Now we return to the two problems which were mentioned at the start of the chapter and discuss their solutions.

### 4.3.1 Getting maximum neighbors

The problem was- given $n$ Voronoi sites in the plane, to find the location where a new Voronoi site can be placed so that it gets the maximum possible number of Voronoi neighbors. We present the solution below.

**Theorem 4.3.1.** *If a new site, p is placed in the cell corresponding to any node in the topmost level of the dual graph of arrangement, then it will get the maximum possible number of existing sites as neighbors.*

*Proof.* As discussed in 4.2.2, the $m^{th}$ level of the dual graph includes the nodes corresponding to the cells of the arrangement of Delaunay circles which were formed by the intersection of $m$ circles and ensures that if a new site is placed inside that region, then it will get $m+2$ neighbors. So, the higher level a node is, the more neighbors it will guarantee. Thus, the topmost level of the dual graph of the arrangement of Delaunay circles contains the nodes which corresponds to the cells in the arrangement which were formed by the intersection of the most number of circles and thus ensures that if the new site is placed inside any of these cells, then it will get the maximum possible number of neighbors. $\square$

**Theorem 4.3.2.** *After constructing the dual graph of the arrangement, the location where a new site must be placed to get maximum number of Voronoi neighbors can be found in constant time.*

*Proof.* According to Theorem 4.3.1, the new site must be placed in a cell whose corresponding node is in the topmost level. And we already mentioned that we can create and update a list of nodes in the topmost level while we are constructing the dual graph of the arrangement. After the dual graph is completed, we can simply choose any node from this list to ensure maximum number of neighbors and tus it takes constant time. $\square$

### 4.3.2 Getting all sites as neighbors

The second problem was, given $n$ Voronoi sites in the plane, to find the minimum number of new sites that has to be placed (as well as the location where they should be placed), so that all the existing $n$ sites become neighbors to at least one of the new sites. The following theorem shows that the problem reduces to the minimum set cover problem.

**Theorem 4.3.3.** *The problem of finding the minimum number of new Voronoi sites needed to be placed to get all the existing Voronoi sites as neighbors to at least one of the new sites reduces to the minimum set cover problem.*

*Proof.* The *minimum set cover problem* is defined as follows

- INSTANCE: Collection $\mathbf{C}$ of subsets of a finite set $\mathbf{S}$.
- SOLUTION: A set cover for $\mathbf{S}$, i.e., a subset $\mathbf{C}' \subseteq \mathbf{C}$ such that every element in $\mathbf{S}$ belongs to at least one member of $\mathbf{C}'$.
- MEASURE: Cardinality of the set cover, i.e., $|\mathbf{C}'|$.

Our problem can be restated in the following way - we have a set of cells, $\mathbf{C}$. And for each cell, $A \in \mathbf{C}$, there is a corresponding set of neighbors ensured by $A$ such that $\mathbf{N}(A) \subseteq \mathbf{N}$, the set of all existing Voronoi vertices. Choosing a cell is equivalent to choosing the set of neighbors ensured by them. Let the set of the set of neighbors be $\mathbf{NC}$. So, each member of $\mathbf{NC}$ is a subset of $\mathbf{N}$. Now our problem can be written as follows.

- INSTANCE: Collection $\mathbf{NC}$ of subsets of a finite set $\mathbf{N}$.

- SOLUTION: A set cover for $\mathbf{N}$, i.e., a subset $\mathbf{NC}' \subseteq \mathbf{NC}$ such that every element in $\mathbf{N}$ belongs to at least one member of $\mathbf{NC}'$.
- MEASURE: Cardinality of the set cover, i.e., $|\mathbf{NC}'|$.

Thus, the problem of finding the minimum number of new Voronoi sites needed to be placed to get all the existing Voronoi sites as neighbors to at least one of the new sites reduces to the minimum set cover problem. □

**Theorem 4.3.4.** *If there are $n$ exisiting sites, then the problem of finding the minimum number of new sites needed to get all of them as neighbors is approximable within* $1 + \ln n$.

*Proof.* Theorem 4.3.3 proved that this problem can be reduced to the minimum set cover problem which is a well-studied $NP-complete$ problem and there is existing algorithm [20] using which it is *approximable* within $1+\ln n$. Hence the proof. □

The following lemma shows how we can restrict the number of sets that need to the considered for the set cover problem.

**Theorem 4.3.5.** *We can restrict the set cover problem to only the sets of neighbors ensured by the cells whose corresponding nodes in the dual graph has no parent.*

*Proof.* For any node $A$, with a parent there is at least one node $B$, such that $\mathbf{N}(A) \subset \mathbf{N}(B)$. So it is sufficient to consider $\mathbf{N}(B)$ and ignore $\mathbf{N}(A)$. Applying this argument to all nodes will eliminate all nodes except the nodes with no parents and thus our problem becomes restricted to sets of neighbors ensured by the cells corresponding to these nodes. □

## 4.4 Summary

In this chapter, we first showed that the number of Voronoi neighbors of a new site is determined by the cell of the arrangement of Delaunay circles in which the site is placed (see Lemma 4.1.3). Then Theorem 4.1.1 established that the set of neighbors ensured by two arc-adjacent cells of the arrangement differs by exactly one element. Based on this theorem we proposed to build the dual graph which can be constructed while the arrangement is being computed using existing algorithms running in $O(n\lambda_4(n))$ time. We also maintain two separate lists at the same time. The first of these lists is a list of the nodes of the dual graph which are in the top level and using this list the problem of finding the location where a new site should be placed to get maximum neighbors can be computed in constant time (see Theorem 4.3.2). The second list contains the nodes which has no parents. We showed that the problem of finding the minimum number of new sites that has to placed to get all existing sites as neighbors is reducable to the minimum set cover problem which is approximable within $1 + lnn$ time (see Theorem 4.3.2) and we also showed that only the sets corresponding to the cells in the second list need to considered for the set cover (see Theorem 4.3.5).

# Chapter 5

# Voronoi neighbor games

In Chapter 3, we have discussed a number of games based on the idea of maximizing the Voronoi area controlled by the sites of each player. We also cited several researches aimed at finding winning strategies for those games. In this chapter we are introducing several variations of Voronoi neighbor games and their solutions. In our knowledge, we are the first to address these games.

In *Voronoi neighbor games*, two players compete against one another with the intention of optimizing their number of Voronoi neighbors. In the rest of this thesis, we shall relate to the players as Player1, who makes the first move and Player2, who makes the last move. Variations arise through the change in the number of rounds played by each player and the exact criterion of winning.

In the rest of this chapter, we shall name the $i^{th}$ site placed by Player1 as $p_i$ and the $i^{th}$ site placed by Player2 as $q_i$ and in the figures the sites placed by Player1 will be represented by solid black circles and the sites placed by Player2 will be represented by solid gray rectangles.

## 5.1 One round games

In *one round games*, as the name suggests, each player gets only one round to place his $n$ sites. In other words, first Player1 will place all $n$ of his sites in 2D, and then Player2 will place his $n$ sites. The game is analyzed for five different winning criteria.

One round games are not perfect information games and is not impartial (refer to Section 2.5). Here, due to its one round nature, Player1 has to place his points without any knowledge whatsoever about how Player2 is going to place his sites. On the other hand, Player2 has complete knowledge of the positions of the sites placed by Player1. This gives Player2 an advantage which enables him to effectively implement winning strategies for almost all the variations.

### 5.1.1 Variation 1: Maximizing opponent neighbors

In this variation, the winning criterion is to get as many distinct opponent sites as neighbors as possible. Before elaborating the winning criterion let us first elaborate the term *distinct neighbors*. Mathematically, if a site $x_i$ of playerX gets $N(x_i)$ number of opponent sites as its neighbor, then the total number of distinct opponent sites as neighbors of PlayerX, $N_x$ is $|\cup_{i=1}^{n} N(x_i)|$.

So, if the number of distinct sites of Player2 which are neighbors of Player1 $= N_1$ and the number of distinct sites of Player1 which are neighbors of Player1 $= N_2$, then the objective of this variation is to maximize $N_1$ and $N_2$. From the criterion it is clear that this is more like a maximization problem, rather than a competition. Now, because Player1 does not know where Player2 will place his sites, it is not possible to formulate a strategy for Player1. On the other hand several strategies can be found for Player2.

**Lemma 5.1.1.** *It is possible for Player2 to get all n sites of Player1 as neighbors.*

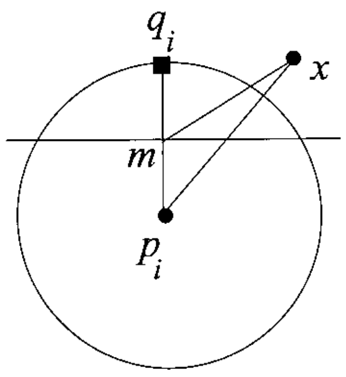


Figure 5.1: Placing a new site close to an existing site

*Proof.* As the game is played on Real space, a site can be placed arbitrarily close to another site. Let, $x$ be the site (can belong to either player) which is closest to a site $p_i$ of Player1 which is not yet a neighbor of any site of Player2. So, Player2 will place his site, $q_i$ in such a way that $p_iq_i < xp_i$. As $p_iq_i$ is smaller than $xp_i$, then $x$ must be outside the circle centered at $p_i$ and has $p_iq_i$ as radius (see Figure 5.1). If $m$ is the point of intersection between $p_iq_i$ and their bisector, then $m$ must be closer to $p_i$ and $q_i$ than $x$. And as $x$ is the closest site to $p_i$, the other sites are also outside the circle and are obviously even further from $m$. This means $m$ is on the Voronoi edge between the Voronoi regions of $p_i$ and $q_i$ and thus $p_i$ and $q_i$ must be Voronoi neighbors.

This way each site of Player2 will get at least one distinct site of Player1 as its neighbor and thus Player2 will get all $n$ sites of player1 as neighbors. □

But quite obviously it is not the best solution. We have already seen that a new site can get many of the existing sites as neighbors. So it should be possible for Player2 to win by using less than $n$ sites of his own. Actually the best winning strategy will be the same as the solution to the problem we addressed in Section 4.3.2. So rather than restating the strategy, here we will establish an upper bound on the number of sites required by Player2.

The upper bound will be established using a constructive algorithm. The basic idea of the algorithm is to use a new site to make three existing consecutive sites $p$, $q$ and $r$ on the hull (the hull might be convex or non-convex) as neighbor by placing the new site inside a cell of the arrangement which ensures all of them as neighbors (refer to Lemmas 4.1.1 and 4.1.2). Before placing the next site, $p$, $q$ and $r$ and their corresponding Delaunay edges are removed from the triangulation and then we choose the another three sites that can be removed from the remaining triangulation. This technique is applied again and again until all sites have become neighbors. Note that removing a site/edge does not mean that the site/edge actually no more exists; but it means that the site/edge will only be considered to be non-existent by the algorithm.

See Figure 5.2 for a few examples of site removal. In Figure 5.2(a) an

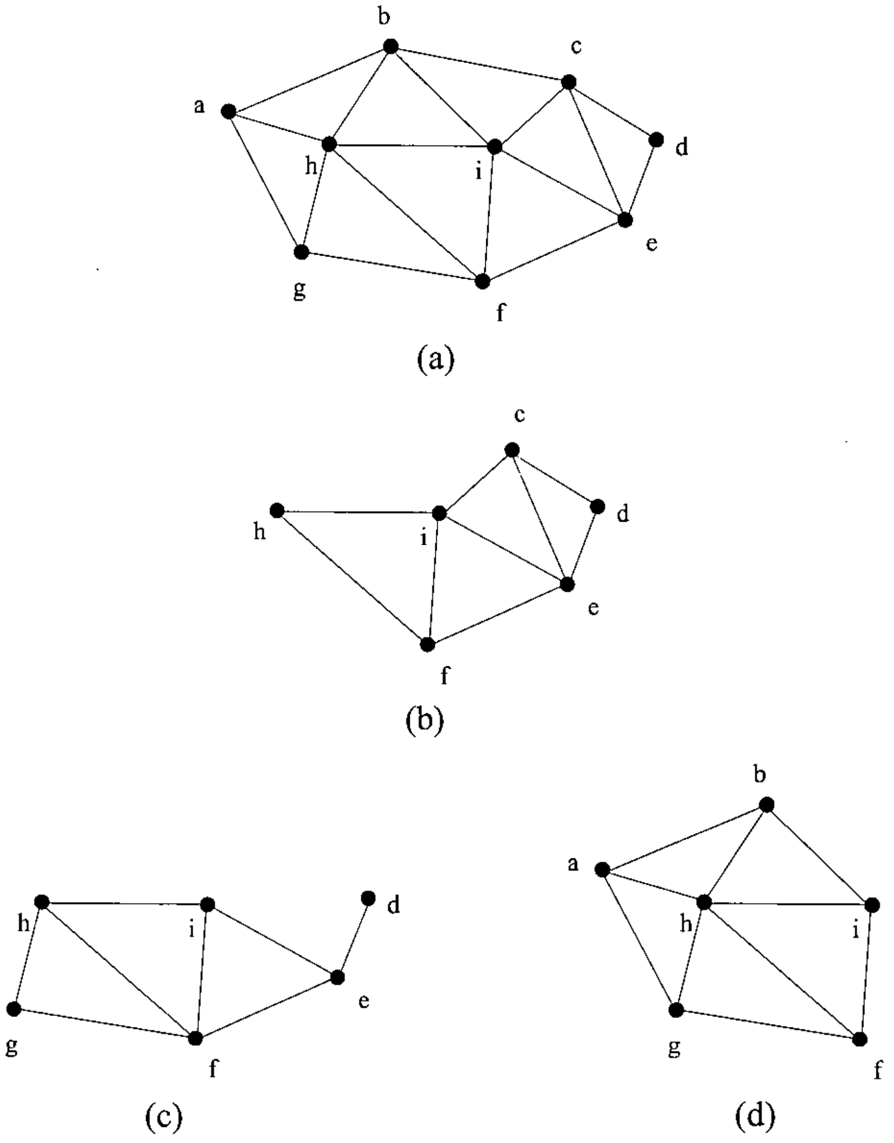


Figure 5.2: Removing three sites from the triangulation

initial triangulation is shown. Figure 5.2(b) shows the remaining triangulation if the sites $g$, $a$ and $b$ are removed. Figure 5.2(c) shows the remaining triangulation if the sites $a$, $b$ and $c$ are removed. Figure 5.2(d) shows the remaining triangulation if the sites $c$, $d$ and $e$ are removed.

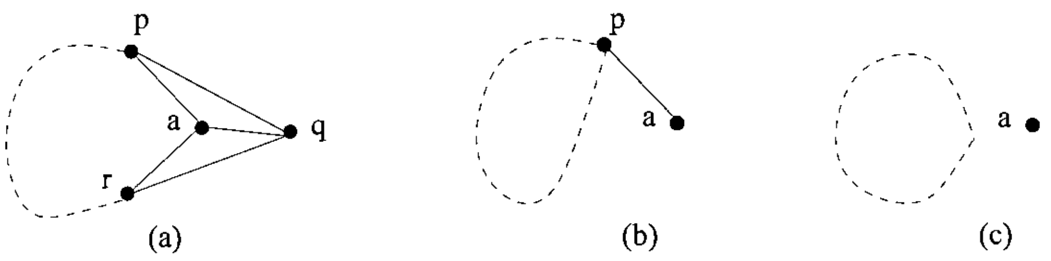


Figure 5.3: Remaining diagram - first problem scenario

From the figures we see that for some combinations of three sites the remaining diagram may not remain a triangulation. [If there is an edge or site which is not part of any triangle, then we say that the remaining diagram is no longer a triangulation]. The following two lemmas identify the cases where such problems arise.

**Lemma 5.1.2.** *If there is a site a, such that it has only three neighbors and at least two of them are selected for removal, then after removal the diagram will not remain a triangulation.*

*Proof.* Let the three neighboring sites of $a$ be $p$, $q$ and $r$ (see Figure 5.3(a)). If two of the neighboring sites $q$ and $r$ are removed, then after removal, $a$ will be on an isolated edge (see Figure 5.3(b)). If all three of the neighboring sites are removed, then after removal, $a$ will not be part of any triangle and will be isolated from the rest of the triangulation (see Figure 5.3(c)). □

**Lemma 5.1.3.** *If any of the three selected sites is part of a triangle such that the other two sites on the triangle are also on the hull (but are not selected for removal), then after removal the diagram will not remain a triangulation.*

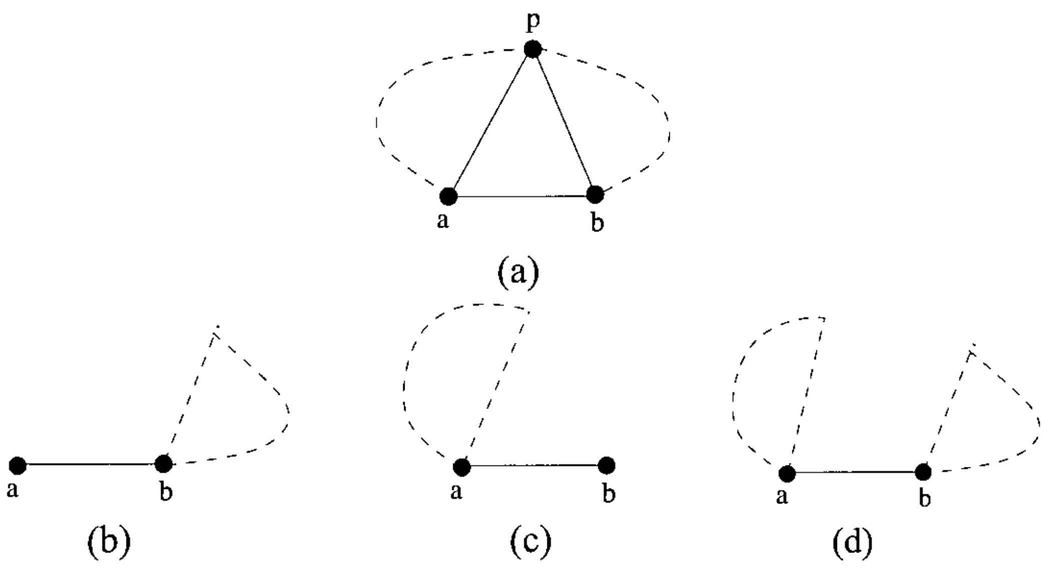


Figure 5.4: Remaining diagram - second problem scenario

*Proof.* Let $p$ be one of the selected sites selected for removal and forms a triangle with two other sites, $a$ and $b$ (none of which are selected for removal) and $ab$ is not part of any other triangle(s) because it is on the hull (see Figure 5.4(a)). Then if we remove $p$ and its adjacent edges, we will be removing $pa$ and $pb$, but not remove $ab$. As $ab$ is not part of any other triangle, it will just be a dangling edge. Depending on the nature of the existing triangulation, the following three outcomes may arise.

- If the triangulation on $pa$'s side contains only the other site(s) selected for removal, then that entire triangulation will be removed and $a$ will become isolated (see Figure 5.4(b)).

- Similarly, if the triangulation on $pb$'s side contains only the other site(s) selected for removal, then that entire triangulation will be removed and

$b$ will become isolated (see Figure 5.4(c)).

- Otherwise, $ab$ will be a connecting edge between two disjoint triangulations (see Figure 5.4(d)).

Whichever be the case, the remaining diagram will not be a triangulation.

□

Though there are some problematic cases, we can still find three sites to remove in almost all cases as shown in the next lemma.

**Lemma 5.1.4.** *It is always possible to find at least three sites that can be removed, except for only one unique situation, such that the remaining diagram is a triangulation.*

*Proof.* We will list the cases where removing a site might be a problem and show that in all such cases, there is an alternate choice available. Let the sites of our concern are $p$, $q$ and $r$ in this order.

- The first problem discussed in Lemma 5.1.2 can be avoided by simply removing the isolated site along with the three selected sites.
- The second problem scenario is if either $p$ or $r$ is part of an ear (except $\Delta pqr$). [If there is a Delaunay triangle such that one of its sites has only two neighbors, then we call this triangle an *ear* of the triangulation]. Let $r$ is part of an ear $\Delta rst$. Now, the following two cases can happen.
  - If $\Delta pqr$ is also an ear, then we can simply remove $q$, $r$ and $s$. See Figure 5.5(a).

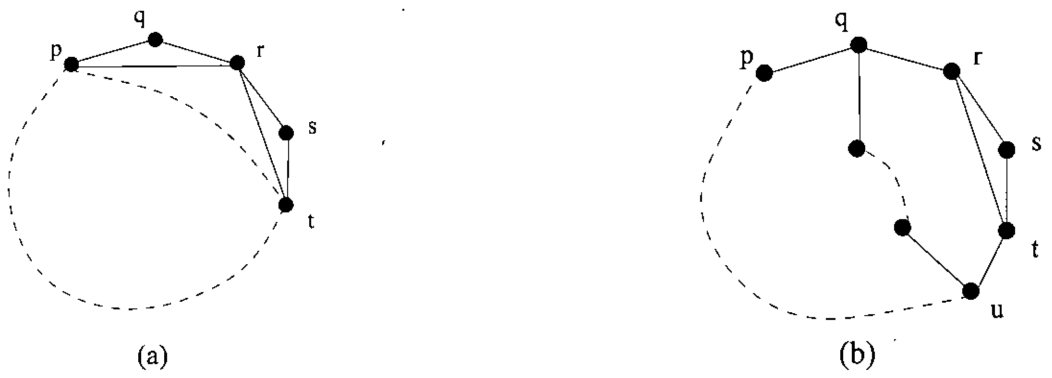


Figure 5.5: Second problem and solution

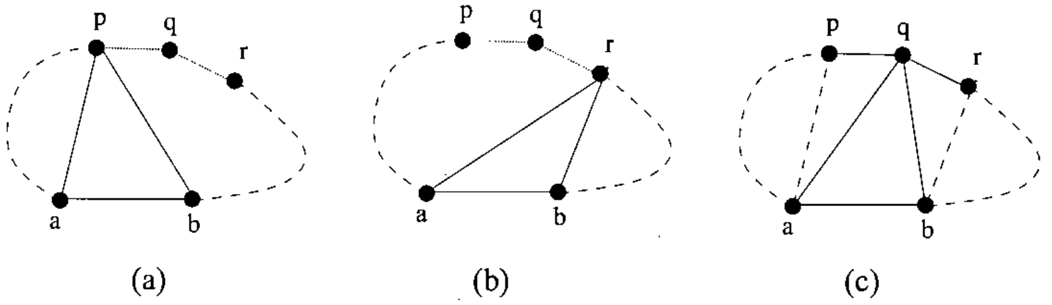


Figure 5.6: Third problem

- If on the other hand, $\Delta pqr$ is not an ear (an thus $q$ is connected to another site inside the hull), then we can remove $r$, $s$ and $t$ (we assume that $t$ is not part of another ear, otherwise it is similar to the first case). See Figure 5.5(b).

- The third problem scenario is if for any one of $p$, $q$ and $r$, there is a triangle with two other sites $a$ and $b$ on the hull and the triangle is not an ear. As, the triangle is not an ear, there must be at least one or more site on the hull on either side of the triangle. If, the problem was

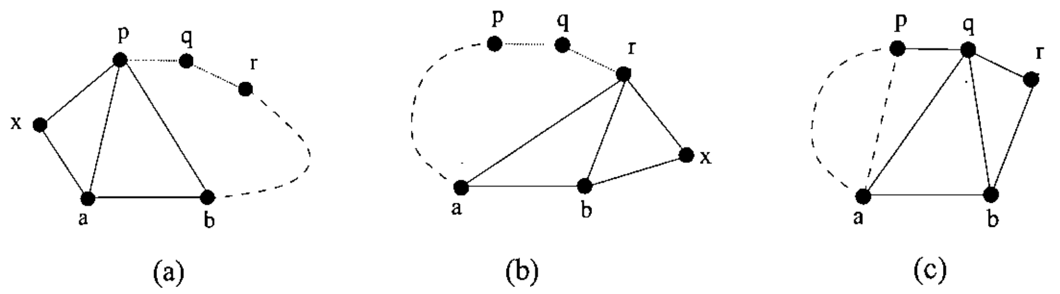


Figure 5.7: Solution of the third problem

with $p$, then there are $q$, $r$, $a$ and possibly more sites on one side of the triangle. Similar situation arise if the problem was with $r$. If the problem was with $q$, then on one side there are $p$, $a$ and zero or more other sites and on the other side there are $q$, $b$ and zero of more other sites. The cases are shown in Figure 5.6.

In each case, we can choose sites from the sites which are on the left or right part of the triangle. If there is again problem with the next sites, then the same argument applies again and in the end when we have only three sites on one side and those three can be selected. In Figure 5.7(a) $p$, $x$ and $a$ can be removed. In Figure 5.7(b) $r$, $x$ and $b$ can be removed. In Figure 5.7(c) $q$, $r$ and $b$ can be removed.

An unique scenario can be found where both the second problem and the third problem are present and it is not possible to remove any three consecutive sites. Here there are two consecutive ears on both side separated by odd numbers of triangle(s) having the third problem. Figure 5.8(a) shows the smallest such example having eleven sites and one triangle having the

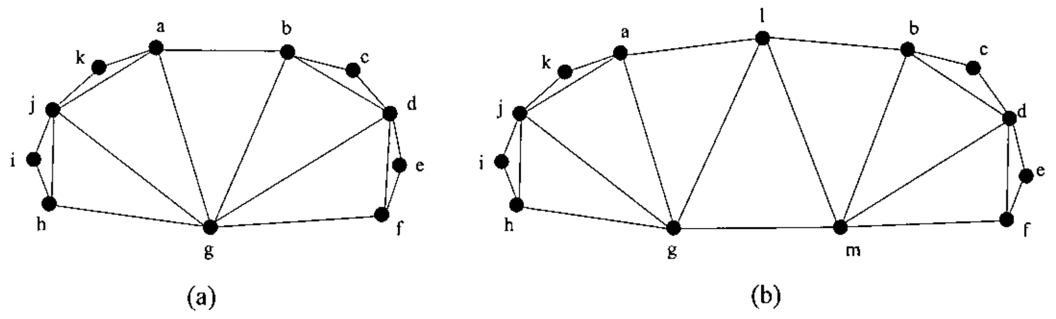


Figure 5.8: An unique problem

third problem. Figure 5.8(b) shows the another example having thirteen sites and three triangles having the third problem. □

**Theorem 5.1.1.** *It is possible for Player2 to get all n sites of Player1 as neighbors using at most $\lceil n/3 \rceil$ sites.*

*Proof.* According to Lemma 5.1.4 Player2 can always place a new site and remove three sites of Player1 in each step. And in the last step (when four or less sites are left), one more new site is placed. Thus it is possible for Player2 to get all $n$ sites of Player1 as neighbors using at most $\lceil n/3 \rceil$ sites.

Even if the special case is encountered, Figure 5.9 shows how four sites are enough to get all eleven remaining sites of Player1 as neighbors. In Figure 5.9(b) we see the outcome if only two sites $e$ and $f$ are removed. After that we can remove three sites in each step as shown in Figure 5.9(c) and 5.9(d). Morever, Player2 could remove three sites of Player1 in each of the steps before encountering this unique scenario. Even if the unique case involved more than eleven sites, we would be able to remove at least three sites in every step except for one. So, overall it is possible for Player2 to get all $n$

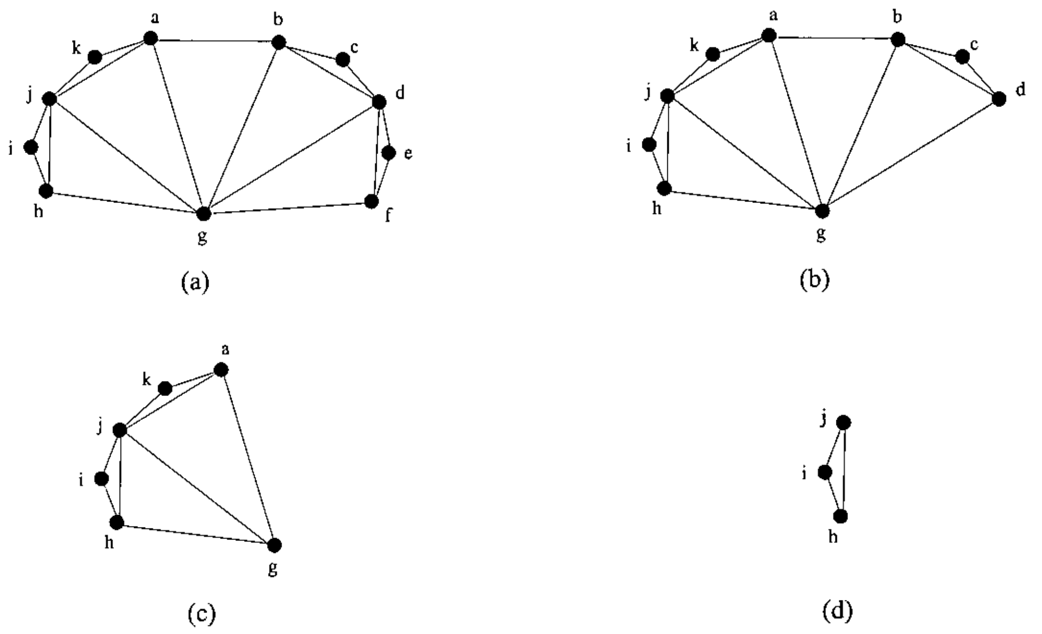


Figure 5.9: Solution to the unique problem

sites of Player1 as neighbors using at most $\lceil n/3 \rceil$ sites. $\square$

It must be mentioned that this is an upper bound on the number of sites required by Player2. The numbers of neighbors gained by each new site mentioned in the above algorithm and proof are the absolute worst case possibilities. The optimal solution (as mentioned in Section 4.3.2) will always use less than or equal to the number of new sites mentioned here.

### 5.1.2 Variation 2: Distinct opponent neighbors

In this variation, the winning criterion is to get more distinct opponent sites as neighbors than the opponent does. In other words, if Player1 gets $N_1$ distinct sites of Player2 as neighbors and Player2 gets $N_2$ distinct sites of Player1 as neighbors, then Player1 wins if $N_1 > N_2$, Player2 wins if $N_2 > N_1$

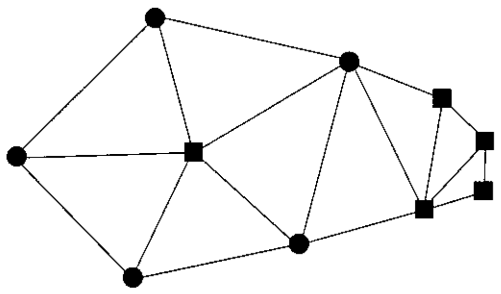


Figure 5.10: Elaboration of the winning criterion Variation 2 of one round games

and it is a tie if $N_1 = N_2$. See Figure 5.10 for further elaboration of the problem. Here, $N_1 = 3$ and $N_2 = 5$, so Player2 wins.

From Theorem 5.1.1, we already know that it is possible for Player2 to get all sites of Player1 as neighbors. So, to win this variation, all Player2 has to do is to ensure that at least one of his sites does not become neighbor to any of the sites of Player1. In the rest of the thesis, this will be referred to as *hiding a site from the opponent.*

**Lemma 5.1.5.** *To hide a site from all the existing sites, two extra new sites are always sufficient.*

*Proof.* First we consider the case when there are at least 4 sites on the convex hull. Let $p$, $q$, $r$ and $s$ be four consecutive sites which are on the convex hull. Now, we will extend the lines $pq$ and $sr$ until they meet at $x$ (if they are parallel, we assume that they meet at infinity) and form the triangle $\Delta qrx$. Now, we will place two new sites, $a$ and $b$ at the midpoints of $qx$ and $rx$ (see Figure 5.11(a)). So, $a$ will become neighbor of $q$ and $b$ will become neighbor of $r$. Again, if $ar < bq$ then $a$ will also become neighbor of $r$; or

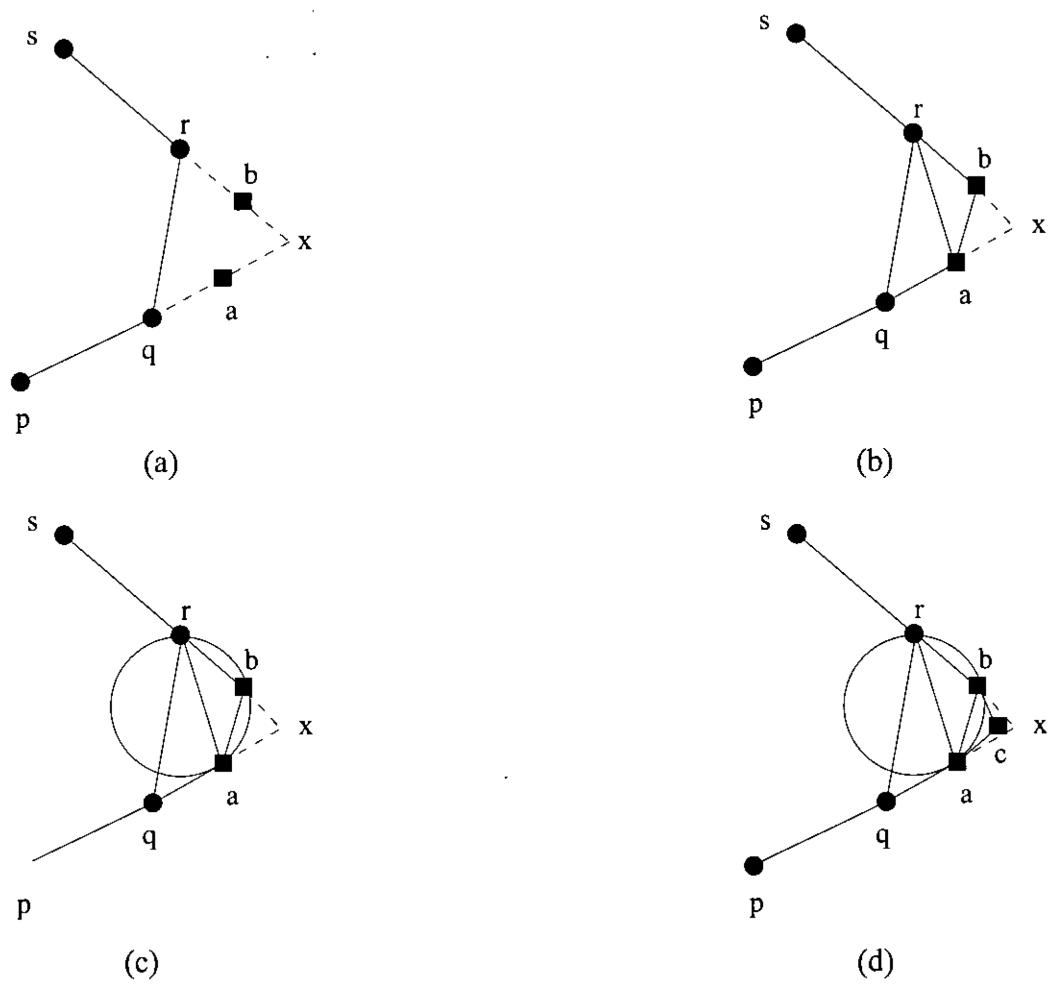


Figure 5.11: Hiding a site

if $ar > bq$ then $b$ will also become neighbor of $q$ [refer to the rule of edge flipping described in Section 4.1.1]. Without loss of generality, let us assume $ar < bq$ (see Figure 5.11(b)).

Now, if another new site, $c$ is placed inside the triangle $\Delta abx$ but outside the circumcircle of $\Delta abr$, then $c$ will be neighbor of $a$ and $b$, but it will not be neighbor of $r$ or, in fact, any other site (see Figure 5.11(c) and Figure

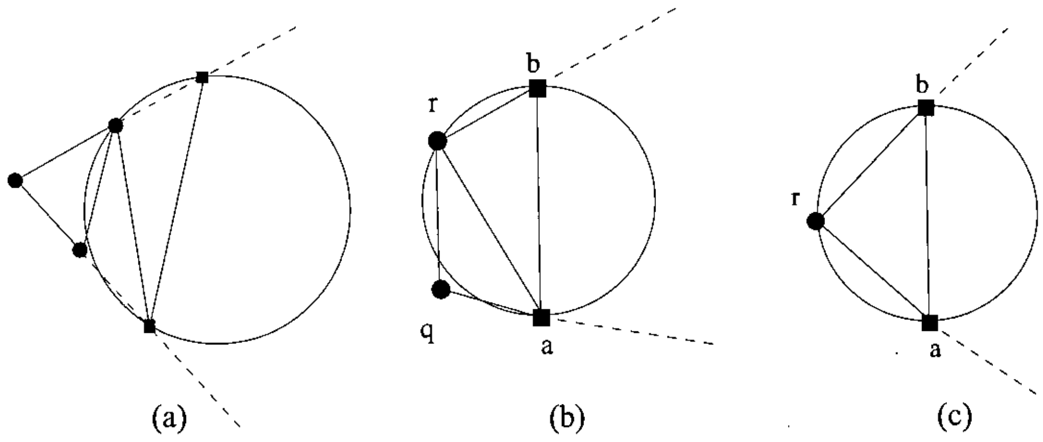


Figure 5.12: Hiding a site when number of sites on the hull is less than four

5.11(d)). Thus, $c$ will be hidden from all the nodes that existed before using the help of two new nodes $a$ and $b$.

When the convex hull consists of less than four sites, then the same technique still suffices. In fact, we get unbounded space to hide the new site(s) after placing the two extra sites, $a$ and $b$. Figures 5.13(a), 5.13(b) and 5.13(c) show how the two extra sites can be placed when there are three, two and one existing site(s) on the convex hull. □

**Theorem 5.1.2.** *Player2 can always get more distinct opponent sites as neighbors than Player1 if $n \geq 3$. For $n < 3$, both player gets equal number of distinct opponent sites as neighbors.*

*Proof.* To win Player2 must complete two tasks- a) get all sites of Player1 as neighbors b) hide at least one site

If $n \geq 5$, then $n - \lceil n/3 \rceil \geq 3$. According to Theorem 5.1.1 Player2 needs at most $\lceil n/3 \rceil$ sites to get all sites of Player1 as neighbors. By Lemma 5.1.5,

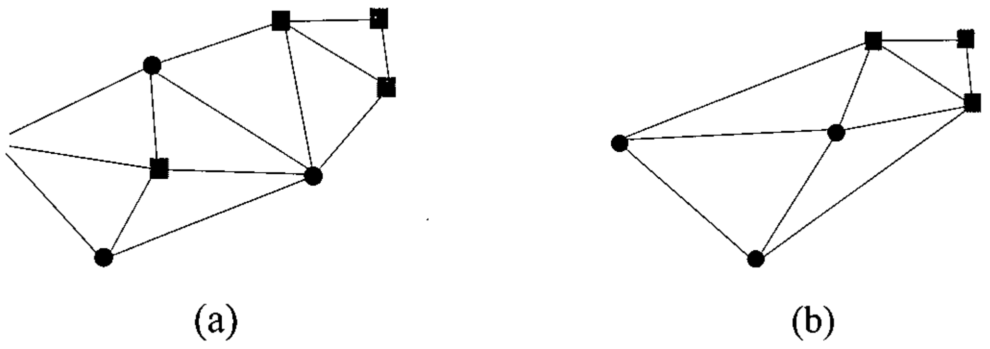


Figure 5.13: One round Variation 2 - for $n$=4 and $n$=3

to hide a site, Player2 needs (2 extra sites plus the site(s) to be hidden). So, if $n \geq 5$ then Player2 wins.

If $n = 4$, then there are exactly two delaunay circles which must intersect each other and thus Player2 needs only one site to get all sites of Player1 as neighbors. Also according to Lemma 5.1.5, because he has three sites left, he can hide one his sites from Player1. Thus Player2 wins. See Figure ??(a).

If $n = 3$, then Player2 can hide one of his sites using two other sites and he can use the same two sites to get all three sites of the opponent as well and thus Player2 can win. See Figure ??(b).

If $n < 3$, Player2 needs only one site to get sites of Player1 as neighbors, but does not have sufficient sites left to hide any sites. So, Player1 also gets all sites of Player2 as neighbors. Thus the game ends in a tie. □

### 5.1.3 Variation 3: Non-distinct opponent neighbors

In this variation, the winning criterion is to get more non-distinct opponent sites as neighbors than the opponent. Before elaborating the winning

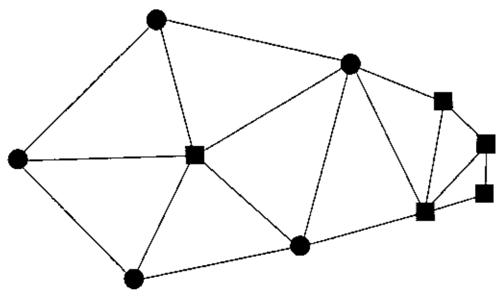


Figure 5.14: Elaboration of the winning criterion Variation 3 of one round games

criterion let us first elaborate *non-distinct neighbors*. Mathematically, if a site $x_i$ of playerX gets $N(x_i)$ many opponent sites as its neighbor, then the total number of non-distinct opponent sites as neighbors of PlayerX, $N_x = \sum_{i=1}^{n} |N(x_i)|$

So, if Player1 gets $N_1$ non-distinct sites of Player2 as neighbors and Player2 gets $N_2$ non-distinct sites of Player1 as neighbors, then Player1 wins if $N_1 > N_2$, Player2 wins if $N_2 > N_1$ and it is a tie if $N_1 = N_2$. See Figure 5.14 for further elaboration of the problem. Here, $N_1 = 8$ and $N_2 = 8$, so it is a tie.

Though it was possible to formulate a winning strategy to get more distinct opponent sites as neighbors (refer to Section 5.1.2), no such strategy can be found when we are counting non-distinct neighbors.

**Theorem 5.1.3.** *No winning strategy exists when the winning criterion is to get more non-distinct opponent sites as neighbors than the opponent does; and the game always ends in a tie.*

*Proof.* Being neighbors is a mutual relationship. If a site, $p_i$, of Player1 gets

$m$ neighbors, then $m$ sites of Player2 will get $p_i$ as their neighbor. Applying the same logic for every site placed by Player2, we can see that the total number of non-distinct neighbors from the opponent sites is the same for both players. □

### 5.1.4 Variation 4: Distinct opponent and self neighbors

In this variation, the winning criterion is to get more distinct (opponent sites as neighbor - self sites as neighbor) than the opponent does. Here, an extra parameter is added to the criterion described in Variation 2 (section 5.1.2). The extra paprameter is the *number of self sites as neighbors.* When a site gets another site belonging to the same player as its neighbor, then they get *self sites as neighbors* or become *self neighbors.*

Now, if $N_1$ is the number of distinct sites of Player2 which are neighbors of Player1, $M_1$ is the number of distinct sites of Player1 which are neighbors of Player1, $N_2$ is the number of distinct sites of Player1 which are neighbors of Player2, and $M_2$ is the number of distinct sites of Player2 which are neighbors of Player2, then the score of Player1, $S_1 = N_1$ - $M_1$ and the score of Player2, $S_1 = N_1$ - $M_1$. So, if $S_1 > S_2$ then Player1 wins; if $S_1 < S_2$ then Player2 wins and otherwise, it is a tie.

Figure 5.15 gives an example. Here, $N_1 = 3$, $M_1 = 5$, $N_1 = 5$ and $M_1 = 4$. So, Player2 wins in this example.

In this variation the two players not only need to maximize opponent sites as neighbors, but also minimize self neighbor ships. So, as well as

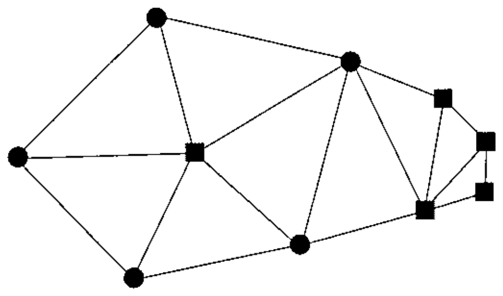


Figure 5.15: Elaboration of the winning criterion Variation 4 of one round games

hiding sites from the opponent Player2 has to try to *hide* his sites from his *own sites* to minimize self neighbor ships. We will see that, the strategy applied for Variation 2, with a small modifition, is sufficient to guarantee that Player2 is the winner for this variation as well.

In the strategy discussed in Variation 2 was to get all $n$ sites of Player1 as neighbors using only $\lceil n/3 \rceil$ sites and then use two sites to hide the rest of the sites from Player1 (see Section 5.1.2). Lemma 5.1.4 showed that we can always find three sites on the hull that can be removed along with the edges incident on them. The technique was to place a new site where it gets the selected three sites as neighbors and then remove the sites and keep repeating the process. Now, we want to add another criteria to the technique. Now we force a restriction on the exact position of the placement.

When Player2 places a new site, $q_i$, to get the selected three sites $p_j$, $p_k$ and $p_l$ of Player1 as neighbors, then it might (or might not) be possible that $q_i$ gets other sites of Player1 as neighbors. It is very easy to check using the dual graph (see Section 4.2). First we shall find the topmost node, $A$, which ensures $p_j$, $p_k$ and $p_l$ as neighbors. If $A$ is in level 1 then no other sites will become neighbors; if $A$ is in level $l$, then Player2 can get at most $l-1$ other sites if he wants (he can get less if he wishes by just choosing a cell corresponding to a lower level node) as well as the three as neighbors using just one new site. We will call these sites *candidates for removal.*

Let $p_h$ is one of the candiates for removal. Now Player2 will check if there is any possibility that the removal of $p_h$ will result in an incomplete triangulation (see Lemma 5.1.2 and Lemma 5.1.3). If the removal does not result in an incomplete triangulation, then $p_h$ will be also be selected for removal along with $p_e$, $p_f$ and $p_g$. Player2 will check all $l-1$ candidate sites in the same way and identify all the sites that can and cannot be removed. Now, he will place the new site such that none of those non-removable sites are neighbors of the new site. Then he will remove the sites that are neighbor of the new site (at least three and maybe more) and their incident Delaunay edges and continue.

The following lemma shows the benifit of this modification.

**Lemma 5.1.6.** *If Player2 places his new site such that all the site of Player1 that becomes its neighbors can be removed, then at most $\lceil n/3 \rceil$ sites used by Player2 will not be neighbors of each other. Morever, each of the sites of Player1 will remain neighbors of at least two other sites of Player1.*

*Proof.* When Player2 places his sites in this way, then after removal none of the remaining sites of Player1 are neighbors of the site placed by Player2. So,

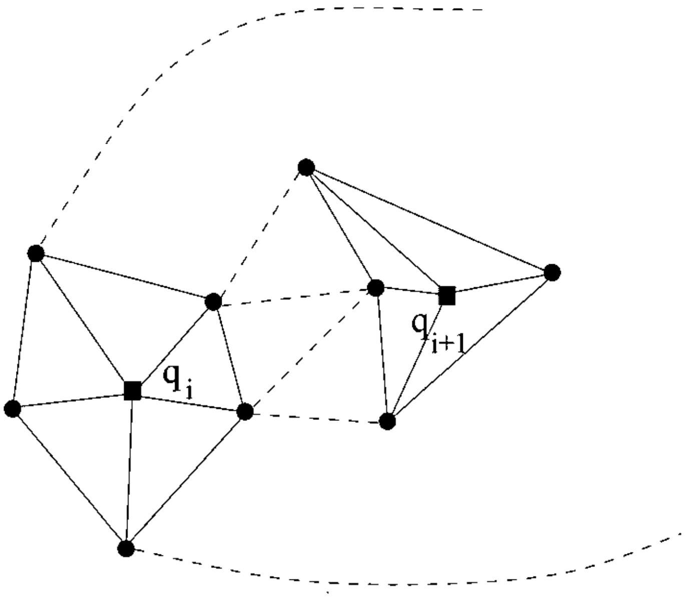


Figure 5.16: Isolated sites of Player1

each site placed by Player2 will be surrounded by sites of Player1 such that clusters having Player1's sites on the boundary and Player2's site as *nucleus* will be separated by the edges (and thus triangles) that were removed. Thus the sites of Player2 cannot be neighbors of each other.

According to the rules of edge flipping (see Section 4.1.1), existing Delaunay edges (i.e. neighborships) are destroyed if they intersect any newly formed Delaunay edge. In this scenario, there can be no Delaunay edges originating at the newly placed site of Player2 and ending at some other site outside the cluster. So the existing Delaunay edges forming the hull of the sites of Player2 which are neighbors of the newly placed site remains intact. Thus the sites of Player1 will remain neighors to at least two other sites of Player1. □

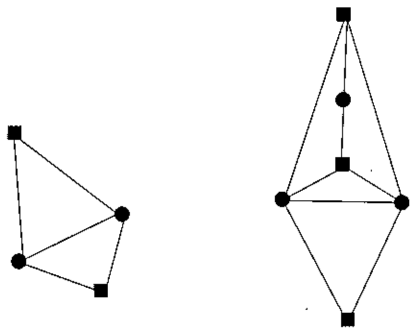


Figure 5.17: Winning Variation 5 of one round game when $n = 2$ and $n = 3$

**Theorem 5.1.4.** *If $n \geq 2$, then Player2 can always get more distinct (opponent sites - self sites) as neighbors than Player1. For $n = 1$, both player gets equal number of distinct opponent sites as neighbors and the game is a tie.*

*Proof.* The second part of the theorem is trivial. Here $N_1 = N_2 = 1$ and $M_1 = M_2 = 0$ and thus the game is a tie.

When $n \geq 4$, according to Theorem 5.1.2, $N_1 \leq (n-1)$ and $N_2 = n$ and, according to Lemma 5.1.6 $M_1 = n$ and $M_2 =$ n - $\lceil n/3 \rceil$. thus, Player2 wins.

For $n = 2$ and $n = 3$, Player2 can place his sites in the way shown in Figure 5.16 and win. □

### 5.1.5 Variation 5: Non-distinct opponent and self neighbors

In this variation, the winning criterion is to get more non-distinct (opponent sites as neighbor - self site as neighbor) than the opponent. More specifically, if $N_1$ is the number of non-distinct sites of Player2 which are neighbors of Player1, $M_1$ is the number of non-distinct sites of Player1 which are neigh-

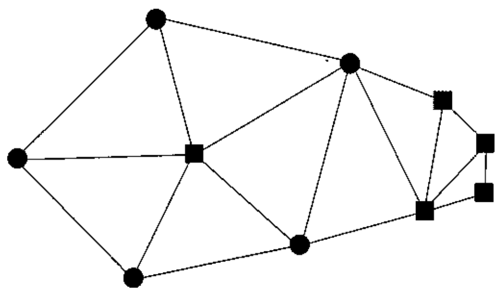


Figure 5.18: Elaboration of the winning criterion Variation 5 of one round games

bors of Player1, $N_2$ is the number of non-distinct sites of Player1 which are neighbors of Player2 and $M_2$ is the number of non-distinct sites of Player2 which are neighbors of Player2, then the score of Player1, $S_1 = N_1 - M_1$ and the score of Player2, $S_1 = N_1 - M_1$. So, if $S_1 > S_2$ then Player1 wins; if $S_1 < S_2$ then Player2 wins and otherwise it is a tie.

Figure 5.18 gives an example. Here, $N_1 = 8$, $M_1 = 10$, $N_1 = 8$ and $M_1 = 10$. So, it is a tie in this example.

Recall from Section 5.1.3 that no winning strategy existed for variation 3 where the criterion was to get more non-distinct opponent sites as neighbors. So, number of non-distinct opponent sites as neighbor is equal for both Player1 and Player2 [i.e. $N_1 = N_2$]. But, the 'number of non-distinct self sites as neighbor' can be different [$M_1$ and $M_2$ can be different] and that difference will decide who wins this variation of the game.

Once again we will try to use the same strategy used before, the 'get all and hide at least one' strategy. But we can easily see that the strategy is not so good for winning this variation. Because the strategy packs a lot of

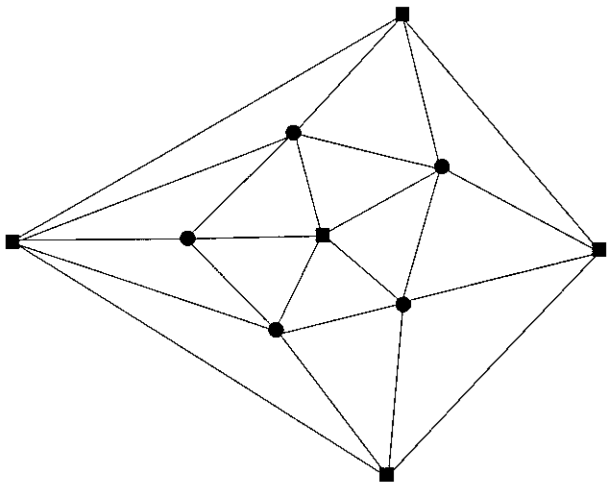


Figure 5.19: Elaboration of *Algorithm 5.1.5.1*

sites of Player2 in a small area (to hide them), the number of non-distinct self neighbors may become quite high and Player2 could lose in many cases. So, a different strategy is warranted. The new strategy is outlined below-

**Algorithm 5.1.5.1**

1. Get all sites of Player1 as neighbors using at most $\lceil n/3 \rceil$ sites following the strategy described in Variation 4.

2. Place the remaining sites uniformly around the outside of the convex hull. The sites should be placed in such a way that they are in convex positions and the distance of each of these sites from their closest site on the hull is greater than the length of their closest edge on the hull

*Algorithm 5.1.5.1* is elaborated using an example in Figure 5.19

**Theorem 5.1.5.** *Algorithm 5.1.5.1 is a winning strategy for variation 5 of one round games.*

*Proof.* Here, we can ignore $N_1$ and $N_2$ as they are equal when we consider non-distinct neighborships.

And, according to Lemma 5.1.6, $M_1 \geq 2n$

Also according to Lemma 5.1.6, the sites of Player2 used for getting all sites of Player1 as neighbors are isolated from each other. But, the rest of the sites of Player2 which are placde outside the hull, will be neighbors of exactly two other sites of Player2. So, $M_2 \leq 2(n - \lceil n/3 \rceil)$

Thus, Player2 wins. □

## 5.2 $n$ Round Games

In *n-round games*, each player gets $n$ rounds to place his $n$ sites. The players take turns to place their sites one at a time inside a 2D region. The next subsection describes the features of $n$-round games, which makes it trickier to formulate a winning strategy.

### 5.2.1 Difference with one round games

In one round games, Player1 had to place his sites without any knowledge whatsoever about how Player2 is going to place his sites. On the other hand, Player2 had complete knowledge of the position of the sites placed by Player1. But in $n$ round games, both players have similar advantages and disadvantages. Here, in the $i^{th}$ round, both players know exactly where the previous $(i-1)$ sites of both Player1 and Player2 were placed.

Thus both players have the ability to dynamically take decisions to optimize each of their turns and both players can actively try to undermine the strategy of the other. For example, it is possible that in the $i^{th}$ round Player2 places a site, $q_i$ and gets a site, $p_i$ of Player1 as its neighbor. But, it is also possible that in the $(i+1)^{th}$ round Player1 places another site, $p_{i+1}$ between $p_i$ and $q_i$, which makes $p_i$ hidden from $q_i$. It is again possible for Player2 to place a site, $q_{i+1}$ in the same round and get both $p_{i+1}$ and $p_i$ as neighbors.

Even if a player makes an optimal move in each step, it does not guarantee that he will win because the other player is also making optimal moves in each round. The only way to win is by taking advantage of either of two unique rounds - the first round or the last round.

There are many examples of games where Player1 can win by making a move in the first round which ensures that no matter how good Player2 plays, Player1 always retains the advantage and in the end Player1 will eventually win. But in our games, such a strategy is not possible because the playing area is virtually unbounded and it does not matter where Player1 places the first site.

The other possibility is for Player2 to make a move in the last round that ensures that he wins. But for Voronoi neighbor games, for most of the variations, the last round is not unique from any other round. If there is a move that can ensure Player2 wins after $n^{th}$ round, then that same strategy should ensure that Player2 is ahead after the $(n-1)^{th}$ round. By repeating the same argument we can say that the same strategy should ensure that Player2 is ahead after every single round. It can either be done by achieving some advantage in the first round and maintaining for the rest of the game or maybe even achieving some advantage in every step. Thus our objective comes down to either finding a unique last move that ensures victory or finding a strategy that ensures that Player2 gets some advantage after the first round and remains ahead after every single round no matter how Player1 plays.

## 5.2.2 Variation 1: Maximizing opponent neighbors

In this variation, the winning criterion is to get as many distinct opponent sites as neighbors as possible. If the number of distinct sites of Player2 which are neighbors of Player1 $= N_1$ and the number of distinct sites of Player1 which are neighbors of Player1 $= N_2$, then the objective of this variation is to maximize $N_1$ and $N_2$.

The strategy in each step is to try to get as many distinct opponent sites as neighbors as possible and at the same time ensuring that the opponent gets as few as possible.

**Theorem 5.2.1.** *In n-round Voronoi neighbor games, if both players makes optimal moves in each round, then after n rounds both player get equal number of distinct opponent sites as neighbors.*

*Proof.* As indicated in the last paragraph of Subsection 5.2.1, we should explore the first round first. After the players place their first site, both have the other as neighbors and none has any advantage (see Figure 5.20).

In the second round, Player1 cannot gain any new sites of Player2 simply because there are no new sites. So, the best Player1 can do is to try to ensure

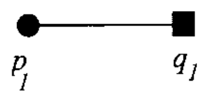


Figure 5.20: State of the game after first round

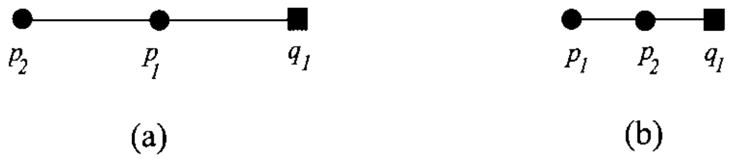


Figure 5.21: State of the game after second move by Player1.

that one of his sites is hidden from Player2 by either placing $p_2$ between $p_1$ and $q_1$ or placing $p_2$ in such a way that $p_1$ is between $p_2$ and $q_1$ (see Figure 5.21 (a) and (b)). If Player1 had placed the site anywhere else, then Player2 would have had 2 opponent sites as neighbors and Player1 would have had only one, giving the advantage to Player2.

Now, if Player2 tries to acquire another site as neighbor by placing $q_2$, then Player1 will also acquire one more neighbor and again none will have any advantage. If it was only a 2-round game then both are winners (see Figure 5.22). On the other hand, if Player2 tried to hide $q_2$, then neither of the players would have gotten optimum number of opponent sites and neither player would have won if it was a 2-round game (see Figure 5.23).

We see that, in each round, Player1 always hides one of his sites from Player2. Player2 can either choose to get the hidden site of Player1 as neighbor or it can hide its own site. For example, all possible outcomes after the end of round 3 is shown in the figures 5.24.

The outcome of each round is equal for both players; i.e. if Player2 decides

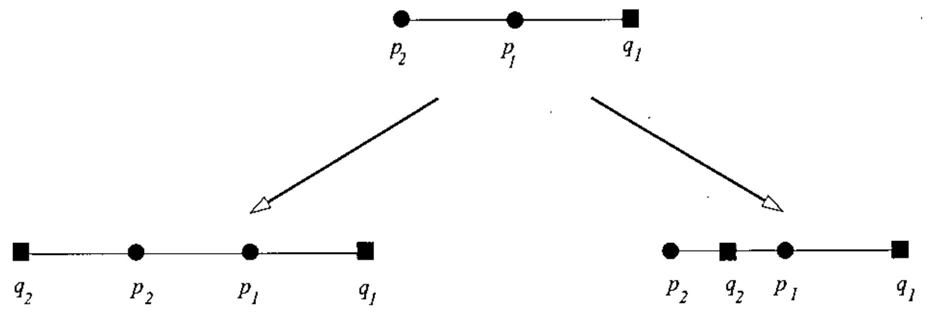


Figure 5.22: State of the game after second round if Player2 attacks.

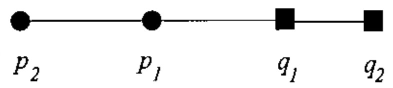


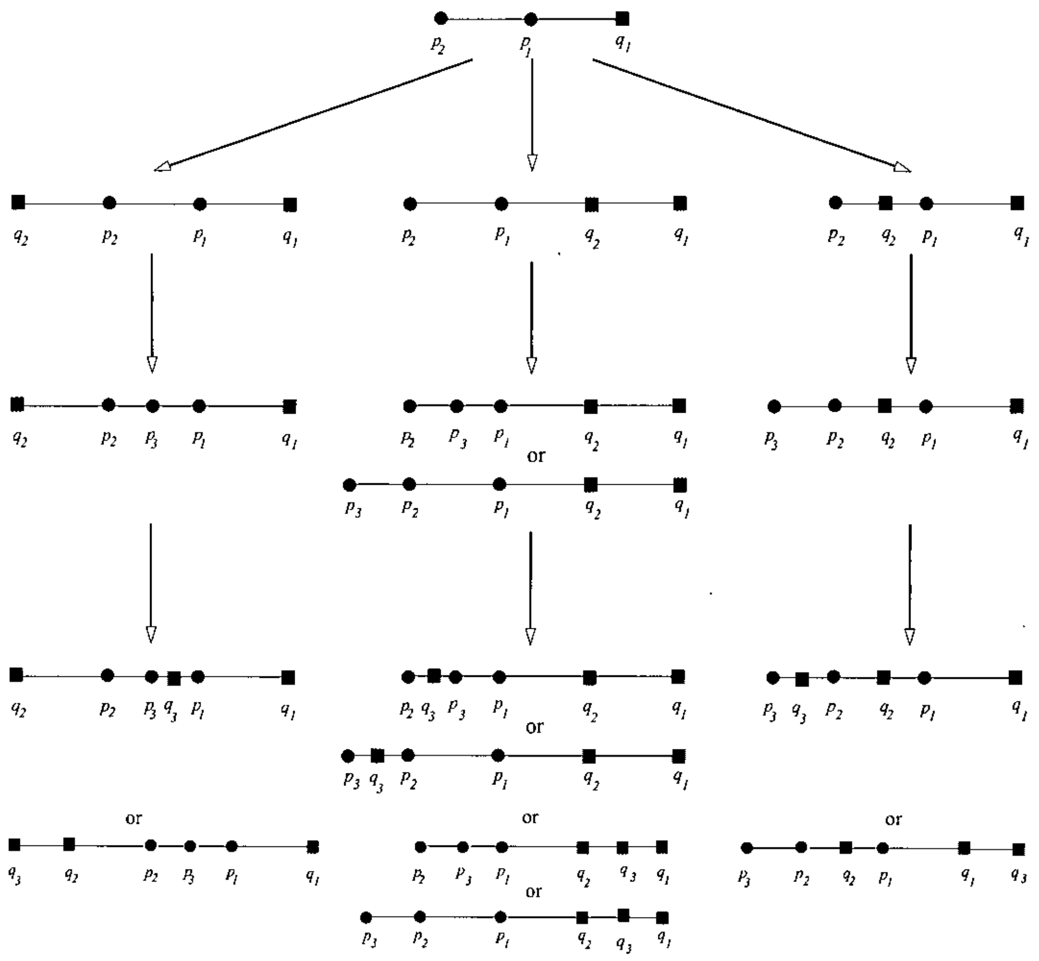
Figure 5.23: State of the game after second round if Player2 hides a site.

to attack, then both players get one more neighbor and if Player2 decides to hide, then neither player gets any new neighbors. Thus, even if the game has more rounds to come, the finally both Players will always have equal number of opponent sites as neighbors. □

### 5.2.3 Variation 2: Distinct opponent neighbors

In this variation, the winning criterion is to get more distinct opponent sites than the opponent. If the number of distinct sites of Player2 which are neighbors of Player1 $= N_1$ and the number of distinct sites of Player1 which are neighbors of Player1 $= N_2$, then Player1 wins if $N_1 > N_2$, Player2 wins if $N_1 < N_2$ and otherwise it is a draw.

The strategy for this variation is actually the same as variation 1 (subsection 5.2.2). And it follows from Theorem 5.2.1 that, this variation ends in a tie if both players play optimally in each round.



$p_2$ $p_1$ $q_1$
$q_2$ $p_2$ $p_1$ $q_1$
$p_2$ $p_1$ $q_2$ $q_1$
$p_2$ $q_2$ $p_1$ $q_1$
$q_2$ $p_2$ $p_3$ $p_1$ $q_1$
$p_2$ $p_3$ $p_1$ $q_2$ $q_1$
or
$p_3$ $p_2$ $p_1$ $q_2$ $q_1$
$p_3$ $p_2$ $q_2$ $p_1$ $q_1$
$q_2$ $p_2$ $p_3$ $q_3$ $p_1$ $q_1$
$p_2$ $q_3$ $p_3$ $p_1$ $q_2$ $q_1$
or
$p_3$ $q_3$ $p_2$ $p_1$ $q_2$ $q_1$
$p_3$ $q_3$ $p_2$ $q_2$ $p_1$ $q_1$
or
$q_3$ $q_2$ $p_2$ $p_3$ $p_1$ $q_1$
or
$p_2$ $p_3$ $p_1$ $q_2$ $q_3$ $q_1$
or
$p_3$ $p_2$ $p_1$ $q_2$ $q_3$ $q_1$
or
$p_3$ $p_2$ $q_2$ $p_1$ $q_1$ $q_3$

Figure 5.24: Possible state of the game after third round

### 5.2.4 Variation 3: Non-distinct opponent neighbors

In this variation, the winning criterion is to get more non-distinct opponent sites as neighbor than the opponent. In other words, if the number of non-distinct sites of Player2 which are neighbors of Player1 = $N_1$ and the number of non-distinct sites of Player1 which are neighbors of Player1 = $N_2$, then Player1 wins if $N_1 > N_2$, Player2 wins if $N_1 < N_2$ and otherwise it is a draw.

The same reasoning applied for variation 3 of one round games (see Subsection 5.1.3) applies here too and as a result the game always ends in a tie regardless of the strategies adopted by the players.

### 5.2.5 Variation 4: Distinct opponent and self neighbors

In this variation, the winning criterion is to get more distinct (opponent sites as neighbor - self site as neighbor) as neighbors than the opponent. Let, $N_1$ = number of distinct sites of Player2 which are neighbors of Player, $M_1$ = number of distinct sites of Player1 which are neighbors of Player1, $N_2$ = number of distinct sites of Player1 which are neighbors of Player2 and $M_2$ = number of distinct sites of Player2 which are neighbors of Player2.

Then we can define the scores of the players as, $S_1 = (N_1 - M_1)$ and $S_2 = (N_2 - M_2)$. The player with a higher score wins.

**Theorem 5.2.2.** *In n-round Voronoi neighbor games when the winning criterion is to get more distinct (opponent sites as neighbor - self site as neighbor) than the opponent and if both players makes optimal moves in each round, the game end in a tie..*

*Proof.* After the first round, $S_1 = S_2 = 1$. So, the game is tied.

Now we consider the second round. Let us analyze the options Player1 has and their consequences in this round. There are three possibilities - a) hiding one of his sites from Player2, b) hiding the new site from his own sites, and c) not hiding. Figure 5.25 shows all three possibilities. Though Player1 is behind in all three cases, but option (b) is the best choice.

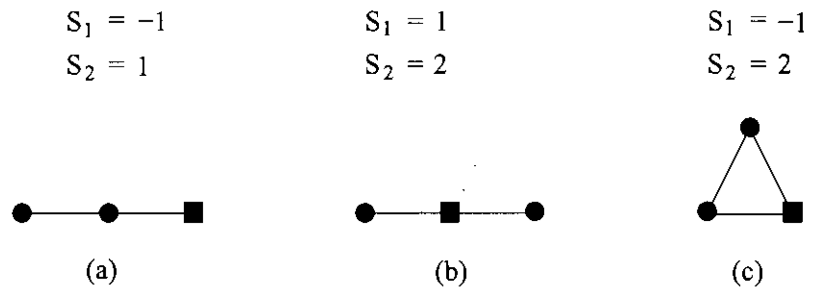


Figure 5.25: State of the game after second move of Player1

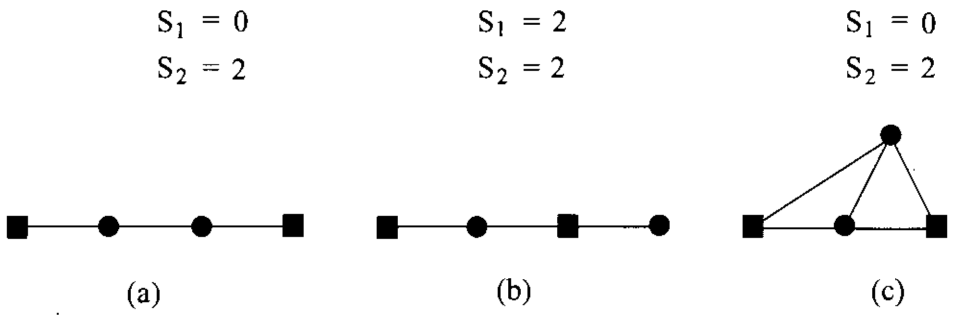


Figure 5.26: State of the game after second move of Player1

Now, Player2's goal is to ensure that his lead is intact or increased in this round. If Player1 chose options (a), then Player2 will use $q_2$ to get as neighbor the site of Player1 which was hidden from him and also try to ensure that $q_2$ is hidden from $q_1$. On the otherhand, if Player1 chose options (b) or (c), then Player2 cannot acquire any new distinct opponent site as neighbors and cannot hide his new site from Player1. So, the best option is to hide $q_2$ from $q_1$. Figure 5.26 shows the state of the game after Player2 places his second site. So, Player2 is ahead unless Player1 chose option (b) in his move.

To generalize Player2's strategy we can state that- if in any round, Player1 uses option (a) and hid one of his site from Player1, then Player2 will use his site to get the hidden site as neighbor and also try to hide the new site from his own sites. If, Player1 used option (b) or (c), then Player2 already has all sites of Player1 as neighbors and the only way to improve is by hiding his new site from his other sites.

Now to find the best policy for Player1 in general, let us assume that, after $(i-1)^{th}$ round, the game is tied. And we can note that, in each round, if Player2 tries to hide his site then Player1 can always get it as neighbor using his new site in the same round. So, after the $(i-1)^{th}$ round, $N_1 = N_2 = i-1$ (due to the strategy followed by Player2) and $M_1 = M_2$ (due to our assumption that the game is tied).

Then, in the $i^{th}$ round, if Player1 chooses option (a), $M_1$ will increase by at least 1, and $N_1$, $N_2$ and $M_2$ remains the same. Then, Player2 can easily get the hidden site as neighbor using $q_i$ and may or may not be able to hide $q_i$ from his own sites. So, $N_2$ and $N_1$ increases by 1, $M_1$ remains the same and $M_2$ may or may not increase. In total, the $i^{th}$ round increases the lead of Player1 or retains it. So, this is not the optimal playing strategy for Player1.

Again, in the $i^{th}$ round, if Player1 chooses option (c), $M_1$ and $N_2$ will increase by at least 1 and $M_2$ and $N_1$ remains the same. Then, Player2 will try to hide $q_i$ from his own sites. So, $N_1$ increases and $M_2$ may or may not increase. In total, the $i^{th}$ round increases the lead of Player2 or retains it. So, this is also not the optimal playing strategy for Player1.

Again, in the $i^{th}$ round, if Player1 chooses option (b), $N_2$ will increase by 1 and $N_1$, $M_1$ and $M_2$ remains the same. Then, Player2 hides $q_i$ from his own sites. So, $N_1$ increases and everything else remains the same. In total, the $i^{th}$ round does not change the outcome of the game.

So, the best strategy for Player1 is to follow option (b) in each step which will ensure that the outcome of the game remains the same. Thus, if both Players follow the optimal strategy the the game ends in a tie. $\square$

### 5.2.6 Variation 5: Non-distinct opponent and self neighbors

In this variation, the winning criterion is to get more non-distinct (opponent sites as neighbor - self site as neighbor) than the opponent. Let, $N_1$ be the number of non-distinct sites of Player2 which are neighbors of Player1, $M_1$ be the number of non-distinct sites of Player1 which are neighbors of Player1, $N_2$ be the number of non-distinct sites of Player1 which are neighbors of Player2 and $M_2$ be the number of non-distinct sites of Player2 which are neighbors of Player2. Then we can define the scores of the players as, $S_1 = (N_1 - M_1)$ and $S_2 = (N_2 - M_2)$. The player with a higher score wins.

**Theorem 5.2.3.** *In n-round Voronoi neighbor games when the winning criterion is to get more non-distinct (opponent sites as neighbor - self site as neighbor) than the opponent and if both players makes optimal moves in each round, the game end in a tie.*

*Proof.* As discussed in Subsection 5.1.5 about variation 5 of one round games, when we count non-distinct neighbors, then $N_1$ will always be equal to $N_2$ and to win a player has to try to reduce $M$. So, in each round, both players

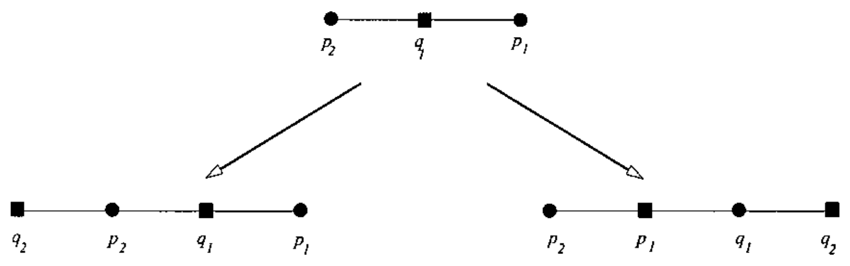


Figure 5.27: After second round

will simply try to hide their new site from their existing sites. This is the optimal strategy.

Once again we will analyze the states of the game starting from the second round (first round is non-consequential). In the second round, Player1 can hide $p_2$ from $p_1$ by ensuring that $q_1$ on the line $p_1p_2$. In the same way, Player2 can hide $q_2$ by ensuring that either $p_1$ or $p_2$ is on the line $q_1q_2$. (see Figure 5.27)

The same pattern will continue and the game will end in a tie. □

# Chapter 6

# Supplementary Results

In this chapter we present few results which, though not integral parts of the primary objective of the thesis, were developed while working with the main objectives and non-trivial enough to deserve a special mention. We will introduce a concept of Voronoi layers and also present a novel algorithm to compute the arrangement of Delaunay circles in linear time. The existing algorithms for computing arrangement of circles takes quadratic time. We show that our algorithm successfully identifies all possible cells of the arrangement generated by the intersection of Delaunay circles which are not more than two Voronoi layers apart. However, our algorithm fails to detect the cells formed by intersection of circles which are more than 2 layers apart. It must be mentioned here that intersections beyond two layers in very rare and it is possible to detect those intersections, but not in linear time. We believe our algorithm gives a very good approximation in quick time and might be useful in many situations.

## 6.1 Voronoi layers

Voronoi layers are defined relatively; i.e. we say that a Voronoi site/region is in layer 2 with respect to a reference Voronoi region. More precisely, Voronoi layers are sets of Voronoi sites/regions such that, Layer 1 consists of the reference site/region, Layer 2 consist of sites/regions that are neighbors of the reference site/region, Layer 3 consist of sites/regions which are neighbors to at least one site/region of Layer 2 but not neighbors of the site/region of Layer 1 and so on. In the rest of the chapter, the statements - 'a site is in layer x' and 'a region is in layer x' are used synonymously.

More precisely, If $p$ is the reference site,

Layer 1, $L_1 = \{p\}$

Layer 2, $L_2 = \{x \mid \forall\, x\ [x \in N(p)]\}$

Layer $n$, $L_n = \{\, x \mid \forall x\, [\, \exists y\, [\, y \in L_{n-1} \,\&\, x \in N(y)] \,\&\, \forall z\, [z \in L_{n-2} \bigcup \cdots \bigcup L_1 \,\&\, x \notin N(z)]]\}$.

Refer to Figure 6.1 for further elaboration. In the figure layers are shown in different shades of gray.

## 6.2 Generating the arrangement of Delaunay circles

We mentioned in Section 2.4 that the arrangement of $n$ circles in the plane can be computed in close to $O(n^2)$ time. But, Delaunay circles have some special

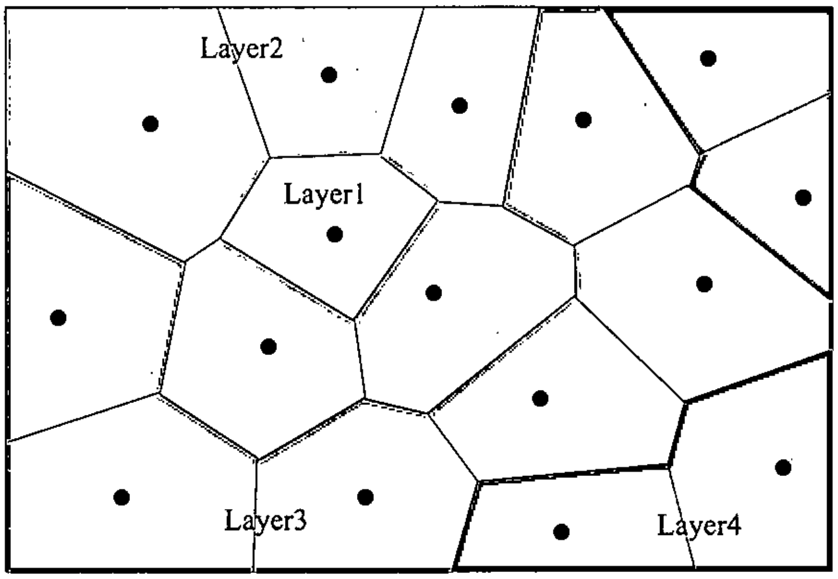


Figure 6.1: Voronoi layers

attributes and properties which might be utilized to develop an algorithm to compute the arrangement which will run faster than the existing algorithm. The following subsections will present the basic idea, the algorithm and its analysis.

### 6.2.1 Basic idea

We know that a property of Delaunay circles is that they always go through three or more Voronoi sites and are centered at the Voronoi vertices (refer to Section 2.1). So, if there are $m$ Voronoi vertices on the perimeter of the Voronoi region of a site $p$, then $m$ Delaunay circles goes through $p$.

Now, if we draw lines connecting the Voronoi vertices to the Voronoi sites, then these lines will actually represent the radii of the Delaunay circles going through that site. We will be able to decide which circles intersect which and

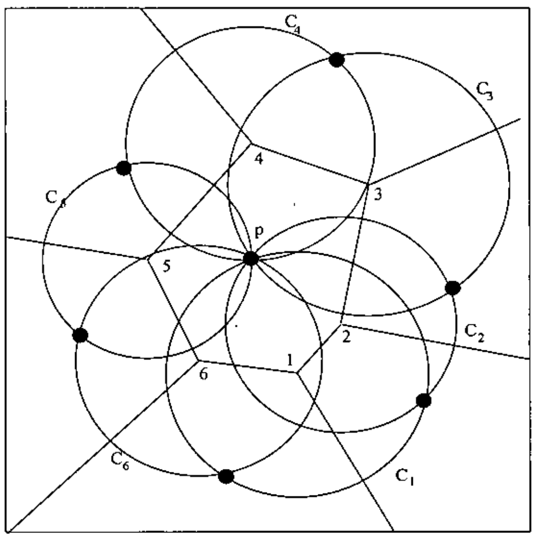


Figure 6.2: Basic idea of generating the arrangement

thus find the cells of the arrangement by simply using the angles between these radii.

### Identifying cells involving two circles

We observe that if more than one circles pass through a single point, then they must either intersect or be tangent to each other. Two circles centered at $c_1$ and $c_2$ are tangent to each other if the line $c_1c_2$ is equal to the sum of their radii. It follows that two circles will be tangent to each other only if the angle between their radii is 180°; otherwise all the circles are pair-wise intersecting and for each intersection there will be a corresponding cell which is part of the two circles.

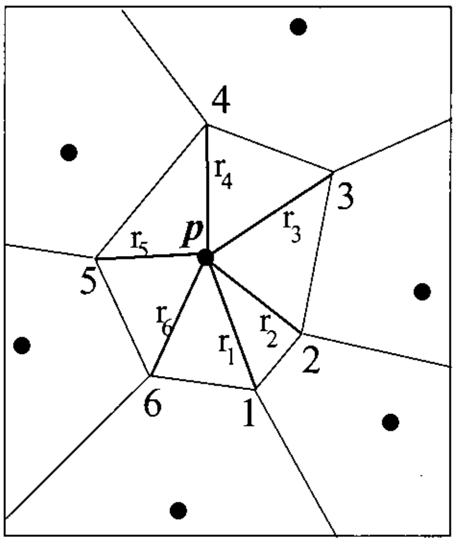


Figure 6.3: The radii of the circles centered round a Voronoi region

### Identifying cells involving three or more circles

A cell involving three circles is formed when two cells each involving two circles, one of which is common, intersect. In other words, if a cell involving circles $A$ and $B$ intersects a cell involving circles $B$ and $C$, then a new cell is formed involving circles $A, B$ and $C$. Such intersections can happen when the sum of the angle between the radii of $A$ and $B$ and the angle between the radii $B$ and $C$ is less than $180°$.

Cells involving more than three circles can also be found by the same technique.

### Going beyond one Voronoi region

The previous subsections elaborated how the intersections of Delaunay circles centered round the perimeter of a single Voronoi region can be found

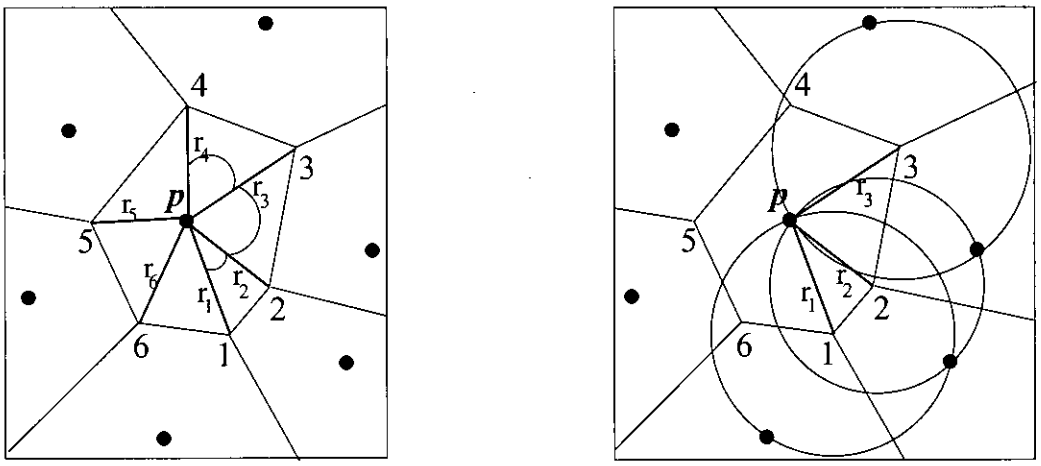


Figure 6.4: Identifying intersections from the radii

out. But each Voronoi vertex is shared by three Voronoi regions and each Delaunay circle passes through three Voronoi sites. So, when we compute the arrangement of the Delaunay circles going through a particular Voronoi site, we only generate a part of all the intersections and cells formed by those circles.

But if we apply the same technique to all the Voronoi regions, then the arrangement will be complete. For example, in Fig 6.5 it is shown that by considering the Regions $V(p)$, $V(q)$ and $V(r)$, the arrangement involving circle $C_1$ is completed.

### 6.2.2 Algorithm

To compute the arrangement generated by the circles centered round a Voronoi site, the algorithm works by rotating an imaginary ray counterclockwise and centred at a Voronoi site. Whenever this ray encounters a radius, it creates a level 1 node for the corresponding circle. It will also create a

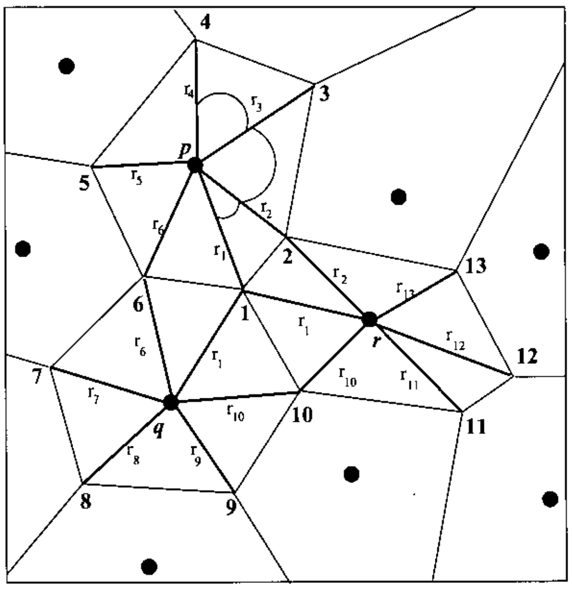


Figure 6.5: Identifying all intersections

level 2 node with the most recent and second most recently created level 1 node as its children. For each node, the algorithm does not store all the circles that formed the corresponding cell, rather it stores the name of one of them- the one whose radius was encountered first. For example, if the algorithm encounters the radii $r_1$, $r_2$, $r_3$ and $r_4$ in this order and there is a node corresponding to the intersection of the circles $C_2$, $C_3$ and $C_4$, then the algorithm will only remember that $r_2$ is the first radius encountered for that node.

The algorithm will continue to create higher level nodes in the same way. But while trying to create a level $i$ node, if it finds that the angle between the radius first encountered for the previously created level $i-1$ node and the current radius encountered by the ray is greater than 180°, then it does not create the level $i$ node or any higher level nodes.

The algorithm terminates when the ray has encountered every radius twice.

**Elaboration of the algorithm**

In this subsection we will elaborate the algorithm with an example. The arrangement for which the algorithm will be applied is shown in Figure 6.6. Figure 6.7 shows the outcome of the algorithm step by step. Figure 6.7(a)-(f) shows the state of the dual graph after the ray encounters 1-6 radii respectively and Figure 6.7(g) shows the final arrangement.

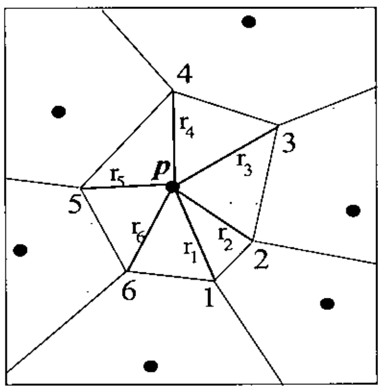


Figure 6.6: The arrangement

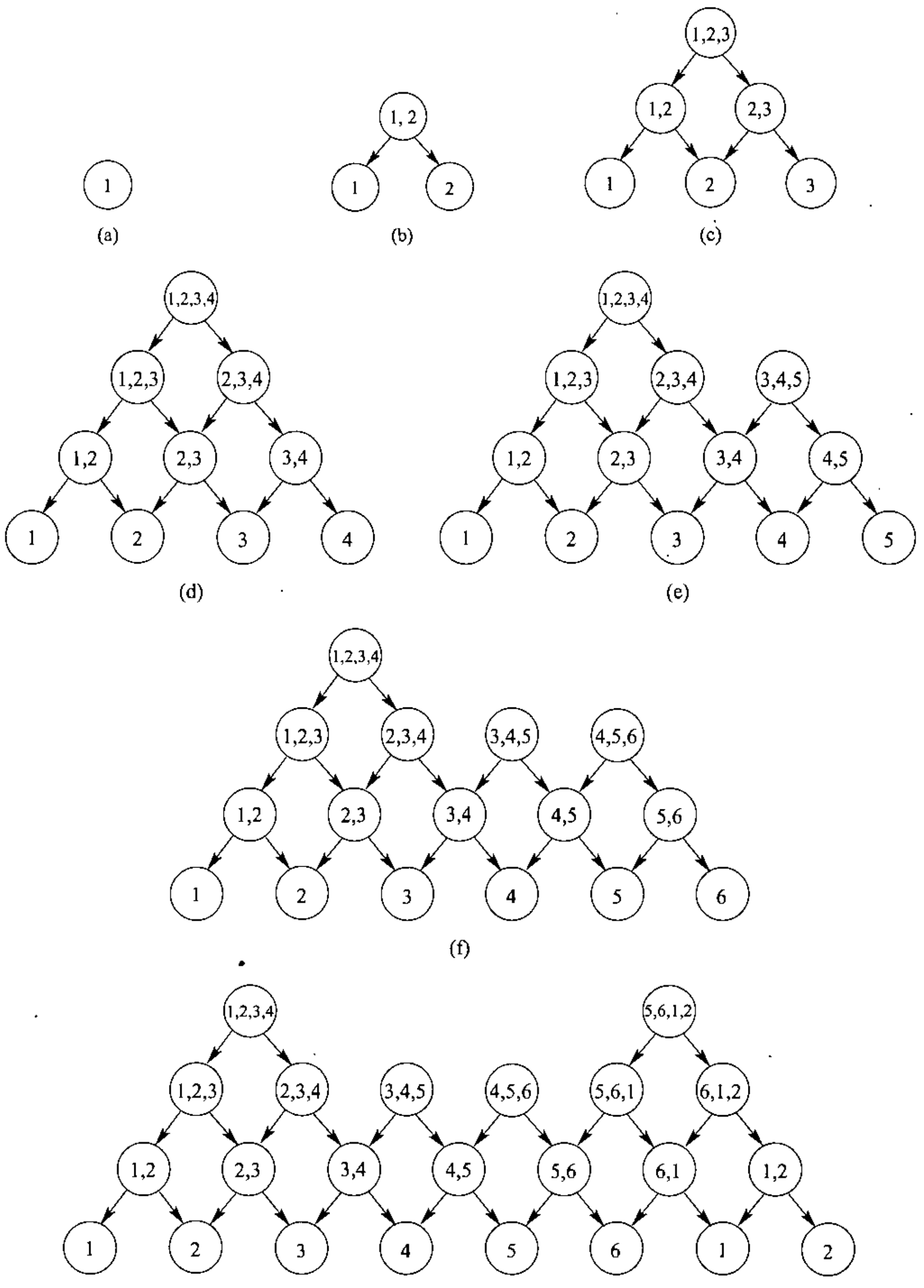


Figure 6.7: Elaboration of the algorithm for constructing the arrangement

### 6.2.3 Complexity of algorithm

The following theorem gives the timing complexity of the algorithm.

**Theorem 6.2.1.** *Our algorithm computes the arrangement in $O(n)$ time*

*Proof.* The ray encounters each radii twice to compute the arrangment for a single Voronoi region. As each Voronoi vertex and thus each radius belongs to three Voronoi regions, so if we consider the entire Voronoi diagram, each vertex will be encountered exactly six times. And as there are at most $2n - 5$ vertices in a Voronoi diagram, the total number of encounters is $12n - 30$.

For each encounter, the number of nodes created is less than or equal to the total number of circles going through a particular Voronoi site. In other words, when we are dealing with $m$ circles, there can be at most $m$ levels. We know that on an average, each Voronoi region has at most 6 vertices (Section 2.1.1). Thus, total complexity of the algorithm is $36(2n - 5)$ or $O(n)$. □

### 6.2.4 Limitation

The limitation of the algorithm presented here is that it cannot detect some possible intersections. It only detects intersections which are within two layer. But in reality there can be intersections beyond layer2. We have given one such example in Figure 6.8.

But we want to mention here that the kind of intersections which are undetected by our algorithm happens very rarely and only when parts of the Voronoi diagram approach degenerate or non-general positions. So we

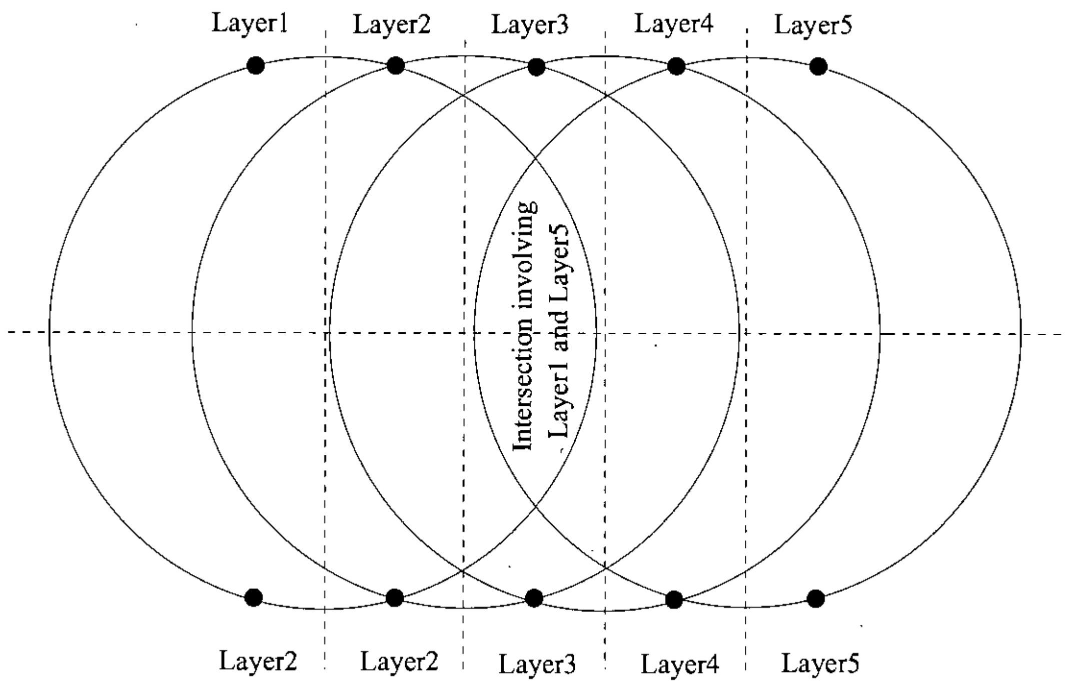


Figure 6.8: Example of intersection beyond 2 layers

believe for many applications (for example, for the one round games) the above algorithm would be sufficient.

# Chapter 7

# Conclusion

In conclusion, we are going to summarize the major contributions of this thesis.

First of all, we have introduced a number of exciting new problems which are both theoretically interesting and have direct applications in facility location. We have established the properties of the arrangement of Delaunay circles, and used them to define an algorithm to find the location where a new site can be placed to maximize its neighbors. Our proposed dual graph of the arrangement can be built while the arrangement is being computed without additional cost and after the dual graph is complete, the location that maximizes Voronoi neighbors can be found out in constant time. We have also proved that the problem of finding minimum number of new sites needed to get all existing points as neighbors can be mapped to the set cover problem, and also proved that the possible candidates for the set cover can be restricted to only the nodes of the dual graph which have no parents.

We are also the first to introduce the Voronoi neighbor games for which we formulated several well defined variations. We devised winning strategies for some of the variations and for the other variations, we proved that the game ends in a tie if both players play optimally and thus no winning strategy is possible.

We also defined an algorithm to construct the arrangement of Delaunay circles in $O(n)$ time and can detect all cells generated by the intersection of Delaunay circles whose centers are at most two Voronoi layers apart. We also identified examples where there can be intersection of circles which are more than two layers apart.

## 7.1 Possibilities for further research

One obvious area for further research is to study the variations of the game for which winning strategies do not exist. If more restrictions are applied on the placement of new sites, then might exist. One example of possible restrictions can be - enforcing that the sites must be placed on a grid and inside a defined boundary. This means that sites cannot be placed arbitrarily close to or arbitrarily far from another site.

Another area where future research can be focused is on improving the algorithm for generating the arrangement of Delaunay circles so that it can detect intersections beyond two layers efficiently.

Also, we proved that the candidate sets for the set cover can be restricted to the sets of neighbors corresponding to the nodes of the dual graph having no parents. Finding the exact number of nodes with this property could be another challenging topic of research.

And finally, in our thesis, we assumed that the service provided by each facility provider were equivalent and the receivers always affiliated with the provider which was geometrically closest. But there can be situations when some service providers can be for preferred by the receivers even if they are geometrically further. In such cases, we can model them using weighted Voronoi diagrams. Therefore, it will be interesting to investigate the maximization and optimization problems as well the games for the weighted Voronoi diagram.

# Bibliography

[1] K. P. Agarwal and M. Sharir. Davenport–schinzel sequences and their geometric applications. Technical report, Durham, NC, USA, 1995.

[2] P. Agarwal, B. Aronov, and M. Sharir. On the complexity of many faces in arrangements of pseudo-segments and of circles. *Discrete and Computational Geometry*, pages 1–23, 2003.

[3] P. Agarwal and M. Sharir. *Arrangements and their applications*, pages 49–119. Elsevier Science Publishers, B.V. North-Holland, Amsterdam, 2000.

[4] H.-K. Ahn, S.-W. Cheng, O. Cheong, M. Golin, and R. van Oostrum. Competitive facility location: the voronoi game. *Theor. Comput. Sci.*, 310(1-3):457–467, 2004.

[5] H.-K. Ahn, S.-W. Cheng, O. Cheong, M. J. Golin, and R. van Oostrum. Competitive facility location along a highway. In *COCOON '01: Proceedings of the 7th Annual International Conference on Computing and Combinatorics*, pages 237–246, London, UK, 2001. Springer-Verlag.

[6] F. Aurenhammer. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[7] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays*, volume 1. Academic Press, 2nd edition, 2001.

[8] C. Chekuri, J. Chuzhoy, L. Lewin-Eytan, J. S. Naor, and A. Orda. Non-cooperative multicast and facility location games. In *EC '06: Proceedings of the 7th ACM conference on Electronic commerce*, pages 72–81, New York, NY, USA, 2006. ACM.

[9] O. Cheong, A. Efrat, and S. Har-Peled. Finding a guard that sees most and a shop that sells most. *Discrete Comput. Geom.*, 37(4):545–563, 2007.

[10] O. Cheong, S. Har-Peled, N. Linial, and J. Matoušek. The one-round voronoi game. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 97–101, New York, NY, USA, 2002. ACM.

[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, 2nd edition, 2000.

[12] F. K. H. A. Dehne, R. Klein, and R. Seidel. Maximizing a voronoi region: The convex case. In *ISAAC '02: Proceedings of the 13th International*

*Symposium on Algorithms and Computation*, pages 624–634, London, UK, 2002. Springer-Verlag.

[13] E. D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *MFCS '01: Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*, pages 18–32, London, UK, 2001. Springer-Verlag.

[14] S. P. Fekete and H. Meijer. The one-round voronoi game replayed. *Comput. Geom. Theory Appl.*, 30(2):81–94, 2005.

[15] S. Fortune. Voronoi diagrams and delaunay triangulations. *Computing in Euclidean Geometry*, 1, 1992.

[16] A. S. Fraenkel. Combinatorial games: Selected bibliography with a succinct gourmet introduction. *Electronic Journal of Combinatorics*.

[17] M. X. Goemans and M. Skutella. Cooperative facility location games. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 76–85, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[18] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi. *ACM Trans. Graph.*, 4(2):74–123, 1985.

[19] M. Hoefer. Non-cooperative facility location and covering games. In *17th International Symposium of Algorithms and Computation, ISAAC 2006*, volume 4288/2006, pages 369–378, Berlin / Heidelberg, 2006. Springer.

[20] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[21] C. L. Lawson. *Software for C1 surface interpolation*, pages 161–194. Academic Press, Orlando, FL, USA, 1977.

[22] T. Moscibroda and R. Wattenhofer. Facility location: distributed approximation. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 108–117, New York, NY, USA, 2005. ACM.

[23] D. B. Shmoys. Approximation algorithms for facility location problems. In *APPROX '00: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 27–33, London, UK, 2000. Springer-Verlag.

[24] D. B. Shmoys, Éva Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 265–274, New York, NY, USA, 1997. ACM.

# Index