

Nearest Facility Location for Multiple Customers using Voronoi Diagram

*Thesis submitted in partial fulfillment of the requirements
for the award of degree of*

**Master of Engineering
in
Computer Science and Engineering**

Submitted By
Ravi Agarwal
(Roll No. 801132032)

Under the supervision of:
Dr. Deepak Garg
Associate Professor



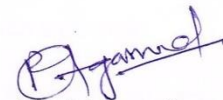
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

July 2013

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "**Nearest Facility Location for Multiple Customers using Voronoi Diagram**", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Deepak Garg* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.


(Ravi Agarwal)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.


(Dr. Deepak Garg)
Associate Professor
Computer Science and
Engineering Department
Thapar University
Patiala

Countersigned by:


(Dr. Maninder Singh)
Associate Professor and Head
Computer Science and Engineering Department
Thapar University
Patiala


(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

I would like to express my deep sense of gratitude towards my supervisor, **Dr. Deepak Garg**, Associate Professor, Computer Science and Engineering Department, Thapar University, Patiala, for his invaluable help and guidance during the course of thesis. I am highly indebt to him for constantly encouraging me by giving his critics on my work. I am grateful to him for giving me the support and confidence that helped me a lot in carrying out the research work in the present form. And for me, it was an honor to work under his guidance.

I also take the opportunity to thank **Dr. Maninder Singh**, Associate Professor and Head, Computer Science and Engineering Department, Thapar University, Patiala, for providing us with the adequate infrastructure in carrying out the research work.

I would also like to thank my parents and friends for their inspiration and ever encouraging moral support, which went a long way in successful completion of my thesis.

Above all, I would like to thank the almighty God for His blessings and for driving me with faith, hope and courage in the thinnest of the times.

Ravi Agarwal

In present world it is hard to survive without new technologies and internet. New technologies and applications have made our life much comfortable and luxurious. Searching for a facility near to many customers is such a problem that even an approximately correct answer can save lot of labor, time and money. The essence of all the facility location problems is to determine the location of the facility and the allocation of the demands of customers, under the condition of the minimum of the cost. This work presents a possible solution for decision makers to reach facility approximately nearer to all customers. In this thesis work an algorithm to tackle nearest facility location for multiple customers has been proposed. It tackles the problem by considering not only the aggregate distances of all customers but also the maximum difference between the farthest customer and nearest customer. It takes time of order $O(n \log n)$ as it is based on voronoi diagram of order $O(n \log n)$ and its own time is of linear order.

The proposed algorithm tackles the problem of finding the nearest facility for multiple customers by considering two criteria. The first one is minimizing the aggregate distances i.e. sum of total distances covered by all the customers. The second one is minimizing the maximum difference i.e. the difference between the farthest customer and the nearest customer. The approach given here uses voronoi diagram construction algorithm as its base algorithm. To find the voronoi diagram of a given space Plane sweep algorithm or Fortune's algorithm named after its inventor is used which computes voronoi diagram in $O(n \log n)$ and is one the most efficient algorithm known for computing voronoi diagram.

The proposed algorithm is tested for various test cases and the result is in accordance with the expected answer for the problem. In future this application can be implemented using hybrid techniques which are the combination of techniques to find the nearest facility. The ultimate goal is to find the nearest facility as per the situation and requirement in a fast and efficient manner.

Table of Contents

Certificate	i
Acknowledgement	ii
Abstract.....	iii
Table of Contents	iv
List of Figures.....	vi
List of Tables	vii
List of Algorithms	vii
Chapter 1 Introduction.....	1
1.1 Motivation	1
1.2 A brief review of voronoi diagrams	2
1.2.1 Definition and basic properties.....	2
1.2.2 Importance of voronoi diagrams	5
1.2.3 Variations of voronoi diagrams	6
1.2.4 Some applications of voronoi diagrams	8
Chapter 2 Literature Review	10
2.1 Various voronoi diagram construction algorithms.....	10
2.1.1 Divide and conquer construction.....	10
2.1.2 Construction by transformation.....	12
2.1.3 Construction through Delaunay triangulation	12
2.1.4 Higher dimensional embedding	13
2.1.5 Incremental construction.....	13
2.1.6 Dynamic construction	15
2.1.7 Parallel construction.....	15
2.1.8 Plane sweep construction	15
2.2 Literature review of facility location problem	16

Chapter 3 Problem statement	22
3.1 Gap analysis and related work	22
3.1.1 Nearest neighbor search.....	22
3.1.2 Minisum facility location.....	23
3.1.3 Minimax facility location.....	23
3.1.4 Maxmin facility location.....	23
3.2 Problem statement.....	23
3.3 Problem formulation	23
Chapter 4 Implementation	25
4.1 Preliminaries of Fortune’s algorithm	25
4.1.1 Background of Fortune’s algorithm.....	25
4.1.2 Data structure for Fortune’s algorithm	28
4.2 Proposed algorithm	31
4.2.1 Algorithm for nearest facility location for multiple customers	31
4.2.2 Explanation of working of proposed algorithm	34
Chapter 5 Testing and results	35
5.1 Test case 1.....	35
5.1.1 Verification for test case 1	40
5.1.2 Result of test case 1.....	40
5.2 Test case 2.....	41
5.2.1 Verification for test case 2	41
5.2.2 Result of test case 2.....	42
5.3 Test case 3.....	43
5.3.1 Verification for test case 3	43
5.3.2 Result of test case 3.....	45
Chapter 6 Conclusion and future scope.....	46
References	48

List of Figures

Figure 1.1 A single cell of voronoi diagram	3
Figure 1.2 A typical voronoi diagram.....	4
Figure 1.3 A largest empty circle of point q with respect to p.....	5
Figure 1.4 Largest empty circle defining the edge and vertex.....	5
Figure 1.5 Classification of voronoi diagram on the basis of different criteria.....	7
Figure 2.1 Divide and conquer approach.....	11
Figure 2.2 Voronoi and its dual Delaunay trianfualtion	12
Figure 2.3 Incremental algorithm in progress.....	14
Figure 4.1 Beach line and sweep line	25
Figure 4.2 Beach line is x-monotone	26
Figure 4.3 Stages of site event	27
Figure 4.4 Flowchart of voronoi diagram construction using Fortune's plane sweep algorithm.....	32
Figure 4.5 Flowchart of proposed algorithm	33
Figure 5.1 Facilities located for test case 1.....	35
Figure 5.2 After one customer added in test case 1	36
Figure 5.3 After two customers added in test case 1	37
Figure 5.4 After three customers added in test case 1	38
Figure 5.5 After four customers added in test case 1.....	39
Figure 5.6 Positions of facilities and customers for test case 2	41
Figure 5.7 Positions of facilities and customers for test case 2	43

List of Tables

Table 4.1 Comparison of various algorithms of voronoi construction	30
Table 5.1 Aggregate distance for test case 1.....	40
Table 5.2 Maximum difference for test case 1	40
Table 5.3 Aggregate distance for test case 2.....	41
Table 5.4 Maximum difference for test case 2	42
Table 5.5 Aggregate distance for test case 3.....	43
Table 5.6 Maximum difference for test case 3	44

List of Algorithms

Algorithm 2.1 Divide and conquer algorithm.....	11
Algorithm 2.2 Incremental algorithm	13
Algorithm 4.1 Voronoi diagram algorithm.....	28
Algorithm 4.2 Handle site event algorithm.....	29
Algorithm 4.3 Handle circle event algorithm	30
Algorithm 4.4 nearest facility location algorithm.....	31

1.1 Motivation

In everyday life people come across many times with situations like finding a place which is near to everyone who wants to gather for particular reason. For example, there are four friends who live in different part of a city. They decided to watch a movie running in all theatres in the city. So they all now plan to meet at a theatre. But the question is which one? Which of the theatre they should select so that it is almost at equal distance from every friend? Two situations arise when finding the solution of this condition. In first case they may come with a solution as a theatre which is nearer to three of them but farther from fourth friend i.e. nearer to many but farther for remaining. No doubt in this way they can minimize the total distance covered by all the friends but at the cost of few friends covering major distance, which is not fair. In that condition the friend farther from selected theatre may decide not to come. But they don't want that their friends take a decision like this. So they want to find the theatre which is at a reasonable distance from every friend's location. In the second case they want to select a theatre so that the difference between distances travelled by any two friends is minimum. But in this case they may select a theatre far from everyone that is at a distance roughly equal for every friend; hence the difference between distances travelled by any two friends is minimized. So this is also not the solution we are looking for.

Hence a solution is required which gives a facility (theatre in above example) which is near to every query point (friends in above example). This thesis work deals with the problem as stated above to find a point (object) in space which is approximately at equal distance from multiple query points. It is discussed how we can apply different approaches to solve this problem and the one which is applied in this work. Here the problem of nearest point of interest to a group of query points is tackled with the use of voronoi diagrams.

The Voronoi diagram is a versatile geometric structure. We have described an application to social geography, but the Voronoi diagram has applications in physics,

astronomy, robotics, and many more fields. It is also closely linked to another important geometric structure, the so-called Delaunay triangulation.

1.2 A brief review of Voronoi Diagram

The name voronoi was coined after the name of a Russian mathematician Georgy Feodosevich Voronoy. He did initial work on voronoi structure. In fact it also been called by other names like Dirichlet tessellations, Wigner-Seitz zones, Thiessen polygons and Domains of actions, most of which are the names of early researchers on this construct in different fields of science.

The Voronoi diagram of n sites in a plane and its dual, the Delaunay triangulation, are considered to be the most important constructions or technique in the area of computational geometry as well as in some other fields of vision and biology, archeology, computer-aided design, chemistry, geography, pattern recognition, physics, etc. Because of practical and theoretical usefulness of Voronoi diagrams (and Delaunay triangulation), their characteristic features, properties as well as algorithms to construct these diagrams are extensively studied and covered in standard text books of the field, and in numerous papers.

1.2.1 Definition and basic properties

Euclidean distance between two points p and q , $\text{dist}(p,q)$, is given by

$$\text{dist}(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$$

Let $P = \{p_1, p_2, \dots, p_n\}$ represents a set of n distinct points in any plane then these points can be considered as sites for voronoi diagram. The voronoi diagram of set $P\{p_1, p_2, \dots, p_n\}$ can be defined as the division of plane into n cells or region, , one for each site in P set, with the property that a point q lies in the cell corresponding to a site p_i if and only if the $\text{dist}(q, p_i) < \text{dist}(q, p_j)$ for each $p_j \in P$ with $j \neq i$. In other words, the voronoi diagram is the division of space in such a way so that any point that lies in the region of a site has this site as the nearest site as compared to others [1]. Sometimes the meaning of voronoi diagram is taken as the subdivision of space showing only vertices and edges. $V(p_i)$ is used to represent the cell of site p_i , said as voronoi cell of p_i . To understand the structure of complete voronoi diagram it is required to study the structure of single voronoi cell first.

- The bisector between two points p and q is defined as the perpendicular bisector of pq . The bisector between two points divides the planes into two

equal halves. The half-plane that contains the p in it is represented by $h(p,q)$ and the half plane that contains the point q in it is represented by $h(q,p)$. A point r lies in the half plane of p iff $\text{dist}(r,p) < \text{dist}(r,q)$.

- A voronoi cell is created as a result of area bounded by the perpendicular bisectors between a site and every other site in the plane. But as an observation it can be showed that not every bisector define the edges of a voronoi cell. In other word a voronoi cell is created by the intersection of the half planes which contains the site in it. This thing can be stated in formal way as

$$V(p_i) = \bigcap_{1 \leq j \leq n, j \neq i} h(p_i, p_j)$$

- According to above property, in extreme case the voronoi cell is made up of n-1 half plane intersection. Thus $V(p_i)$ is a region constructed as the intersection of n-1 half planes. Hence it is convex in shape that is bounded by at most (n-1) vertices and at most (n-1) edges.

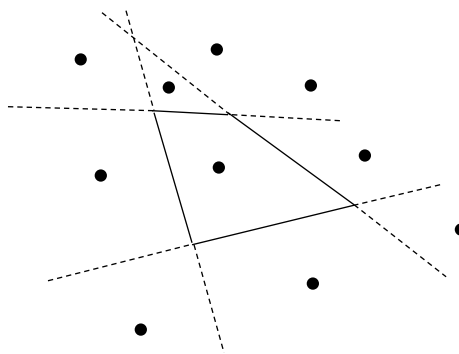


Figure 1.1 A single cell of voronoi diagram

- Every edge in voronoi diagram is a straight line. It may be a line segment or half line (bounded from one side and free from other). Sometimes infinite lines also represent an edge of a voronoi diagram but that is a special case where all the sites are collinear and other edges are also infinite lines.
- It can be put in the form of formal statement as property of voronoi diagram: Let $P\{p_1, p_2, \dots, p_n\}$ be a set of points in the plane representing as sites. The voronoi diagram will have all the edges as infinite parallel edges if all the sites

are collinear otherwise the voronoi diagram will have one (in case when there are only three sites) or more than one intersection point (vertices) and the edges will be either line segments or half infinite line. In this case the voronoi diagram will be a connected diagram.

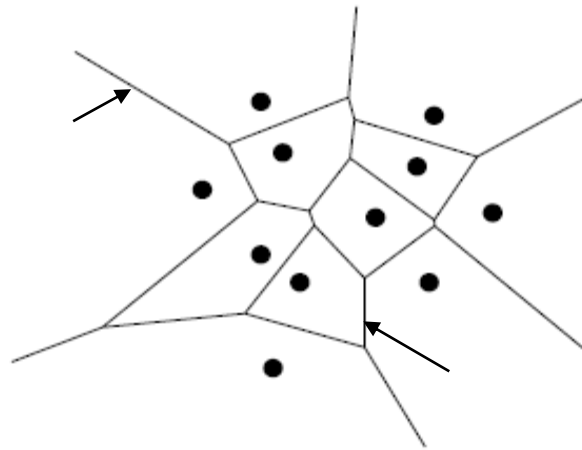


Figure 1.2 A typical vornoi diagram

- In a voronoi diagram, for sites(n) ≥ 3 the number of vertices is at most $2n-5$ and the number of edges is at most $3n-6$.
- Edges are the part of perpendicular bisectors between two sites. The numbers of bisectors are quadratic in nature but the complexity of voronoi diagram is linear. Hence not all the bisectors define the edges of the voronoi diagram and hence not all the intersection of these bisectors defines the vertices of voronoi diagram.
- To characterize which perpendicular bisectors contribute in defining the edges and vertices of a voronoi diagram the concept of largest empty circle is used.
- For any point q in voronoi diagram, if a circle of maximum size is drawn by taking q as its centre so that the circle does not contain any site inside it, is called as the largest empty circle of point q in P (set of sites), denoted as $C_P(q)$.

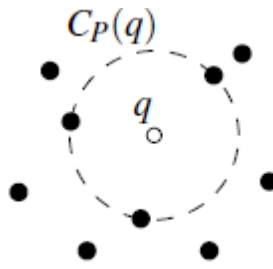


Figure 1.3 A largest empty circle of point q with respect to P

- A point q is a vertex of voronoi diagram $\text{Vor}(P)$ if and only if the largest empty circle of q with respect to P , $C_P(q)$ contains three or more than three sites on its circumference.
- The perpendicular bisector between two sites define an edge if and only if there is a point on the bisector such that the largest empty circle of q with respect to P , $C_P(q)$ contains only these two sites on its circumference and no other site.

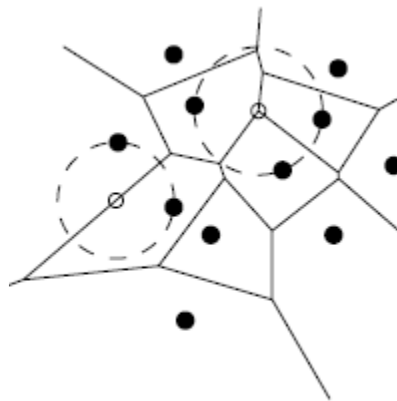


Figure 1.4 Largest empty circle defining the edge and vertex

1.2.2 Importance of voronoi diagram

Voronoi diagram is a very important construct because of the following points:

- Voronoi diagram is found in nature in various situations and various forms. Various types of voronoi diagrams can be used to define the pattern found in nature.

- Voronoi diagrams resembles many structure found in nature and are also related to various well known geometrical structure. Hence voronoi diagrams have interesting geometrical and mathematical properties which can be used to solve mystery of nature.
- Voronoi diagram can be used to solve various complex computational problems when used as data structure. It is used as strong tool to solve many computational problems specially related to geometric structures.
- If any geometric problem can be molded in the form of voronoi diagram, then its many characteristics and properties can be found by simulating it in the form of voronoi diagram.
- Hence by considering above points it could be said that voronoi diagrams are useful in solving problems related to field of mathematics, geometry, natural science and algorithms.

1.2.3 Variations of voronoi diagrams

Voronoi diagrams have emerged from the very basic voronoi diagram to highly complex in nature to solve required complex problems. From the beginning of its invention and use its now have covered diversified fields even those which seems to be not related to computational geometry.

The major varieties of voronoi diagram are stated in following section [2].

(a) Based on order (k)

In this variation of voronoi diagram, regions of voronoi diagram are defined by a subset P_i containing k sites from whole set of sites P [3]. Every point that is nearer to these k sites as compared with other remaining sites lie in a region.

A point q lies in a region of P_i if and only if $d(q,a) < d(q,b)$ for each a in the set P_i and for each b in the set $(P-P_i)$. When the value of $k=1$ the voronoi diagram becomes simple point voronoi diagram where regions are formed on the basis of locality of a single point.

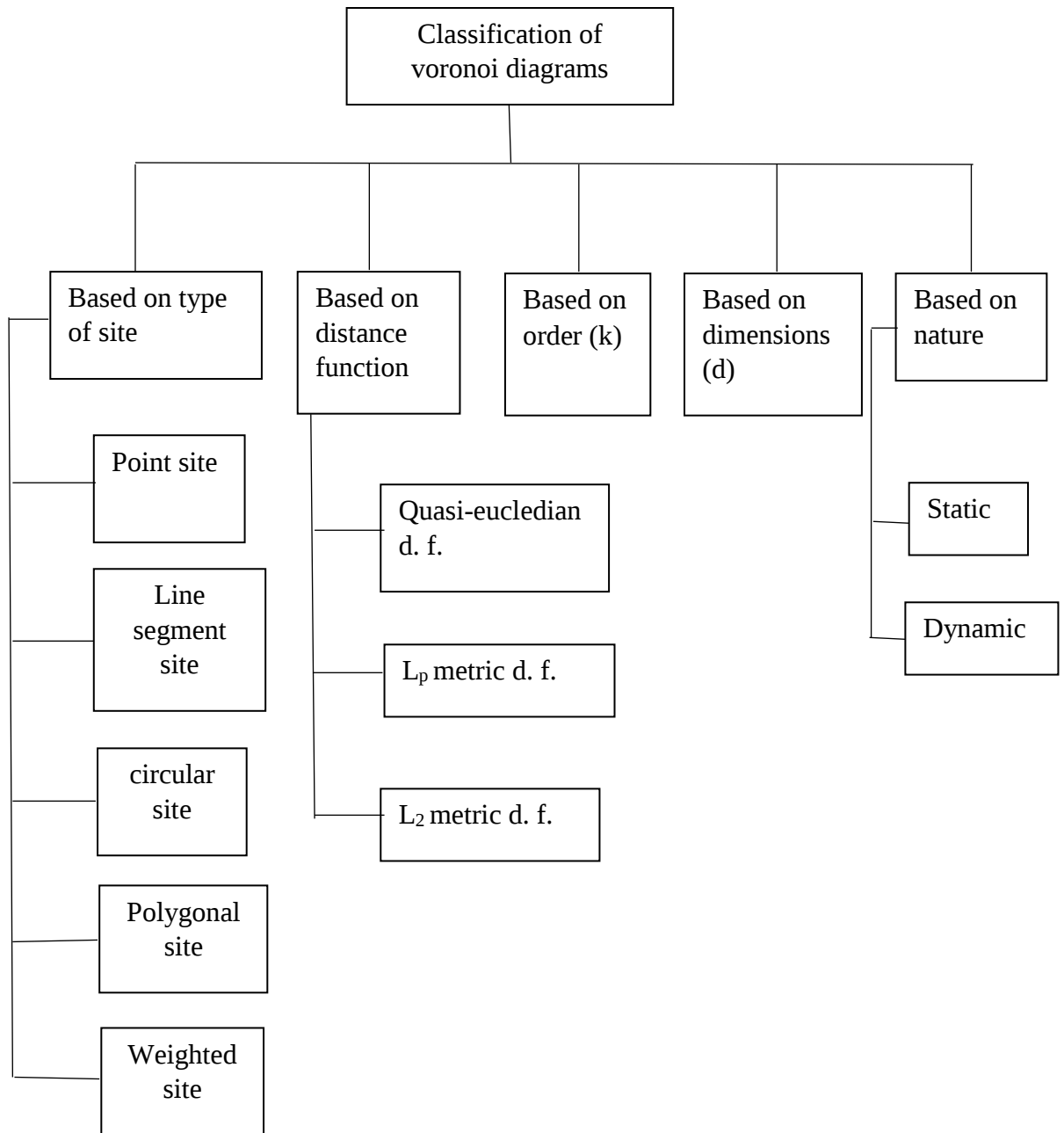


Figure 1.5 Classification of voronoi diagrams on the basis of different criteria

(b) Based on dimensions (d)

Voronoi diagram can be categorized on the basis of the dimensions in which it is defined.

For defining voronoi diagram above two dimensions the concept of facets and edges is used. It is used to represent real world objects and other more complex concept.

(c) Based on shape of sites

- **Point site:** when the shape of sites is point.
- **Line segment site:** when instead of point a line segment defines the site and its region.
- **Circular site:** when site is in the form of a circle. This type of voronoi diagram is used in robot navigation in proximity detection.
- **Polygonal sites:** in this type of voronoi diagram connected polygon are treated as the sites.
- **Weighted site:** in this type of voronoi diagram each site may have different weight assigned to them based on the contribution of the sites in the set.

(d) Based on distance function

- **Quasi-euclidian distance function:** here distance is measured in terms of quasi-euclidian distance.
- **L_p metric distance function:** here L_p metric is used as a distance function.
- **L_2 metric distance function:** here L_2 metric is used as a distance function.

(e) Based on nature

- **Static:** in this type of voronoi diagram sites are known in advance and hence information is available before the process of computing voronoi diagram.
- **Dynamic:** in this type of voronoi diagram, either sites are in constant motion or the sites are added or deleted dynamically.

1.2.4 Some applications of Voronoi Diagram:

- In the field of geometry, voronoi diagrams can be used to find the appropriate location to open new branch of any business as far from others as possible by using the largest empty circle property of voronoi diagram.
- Voronoi diagrams coupled with the farthest point voronoi diagram are used for finding efficient algorithms to compute roundness of a set of points.
- In the field of networking voronoi diagrams can be used to calculate the capacity of a wireless network and construction of Wi-Fi radio map [4].

- In the field of epidemiology, Voronoi diagrams can be used to correlate many sources of infections in epidemics. Such use of voronoi diagram is mentioned in the history by John to study the 1854 Broad Street cholera outbreak in Soho, England. He correlated areas on the map of London using a particular water pump and the areas where most deaths due to the outbreak occurred.
- Voronoi diagrams, known as Thiessen polygons in the field of climatology are used to calculate the overall rainfall based on the series of point measurements.
- Finding nearest neighbor problems are related to many other problems and application, such as finding the nearest hospital or chemist or nearest police station. To answer nearest neighbor queries, voronoi diagram can be used as a point location data structure.
- In the field of ecology the voronoi diagrams are used to study the growth pattern of forest. It can prove useful to prevent fire by building some preventive measures to predict the fire.
- The voronoi diagram can be used to assess the circularity or roundness of the dataset with the help of coordinate measuring machine.
- In the field of polymer physics, voronoi diagram are used to represent free volumes of polymers so that a quality polymer can be made.
- In the field of computational chemistry, Voronoi cells defining the positions of the nuclei in a molecule are used to compute atomic charges. Voronoi deformation density method is applied here.
- In the field of mining, voronoi diagrams are used to find the reserves of very useful minerals and other materials. In this technique exploratory drill holes represents the set of sites in the voronoi diagram.
- In the field of materials science, polycrystalline microstructures in metallic alloys are commonly represented using Voronoi tessellations.
- In computer graphics, Voronoi diagrams are used to generate some specific set of texture like organic or natural substance.
- In the field of robotics, to find the path of a robot navigating through many obstacles [5].

2.1 Various voronoi diagram construction algorithms

The existence of different types and uses of Voronoi diagrams requires their computer construction to vary accordingly. This section reviews methods for the computer construction and representation of planar Voronoi diagrams which have been designed and/or implemented before.

2.1.1 Divide and Conquer Construction

A most common approach that is used to solve many problems, divide and conquer is also applied to construct voronoi diagram. In this methodology the given n sites are divided into two sets by a vertical line and then this approach is applied on recursively until single sites are left. Then after computing diagram for small number of sites merging steps occur resulting in a final diagram of voronoi. It is calculated that a merging process takes $O(n)$ time. Hence full voronoi diagram can be computed in $O(n \log n)$. This way voronoi diagram can be computed in efficient way.

Let P be a set of n points in the plane. The points are vertically partitioned into two subsets R (red) and B (blue). Consider the Voronoi diagram of the sets R and B . Then the Voronoi diagram of P substantially coincides with the Voronoi diagrams of R and B . In fact, there exists a monotone chain of edges of $\text{Vor}(P)$ such that $\text{Vor}(P)$ coincides with $\text{Vor}(R)$ to the left of the chain, and it coincides with $\text{Vor}(B)$ to its right.

Let $b(R, B)$ be the set of all edges and vertices of $\text{Vor}(P)$ belonging to the common boundary of the regions of some $p_i \in R$ and $p_j \in B$. The bisector $b(R, B)$ contains two half-lines, belonging to the bisectors b_{ij} of the two “bridges” connecting the convex hulls of R and B . The bisector $b(R, B)$ is a y -monotone chain leaving the regions of the points $p_i \in R$ to its left and those of $p_j \in B$ to its right. Let R and B respectively be the regions of the plane located to the left and to the right of $b(R, B)$.

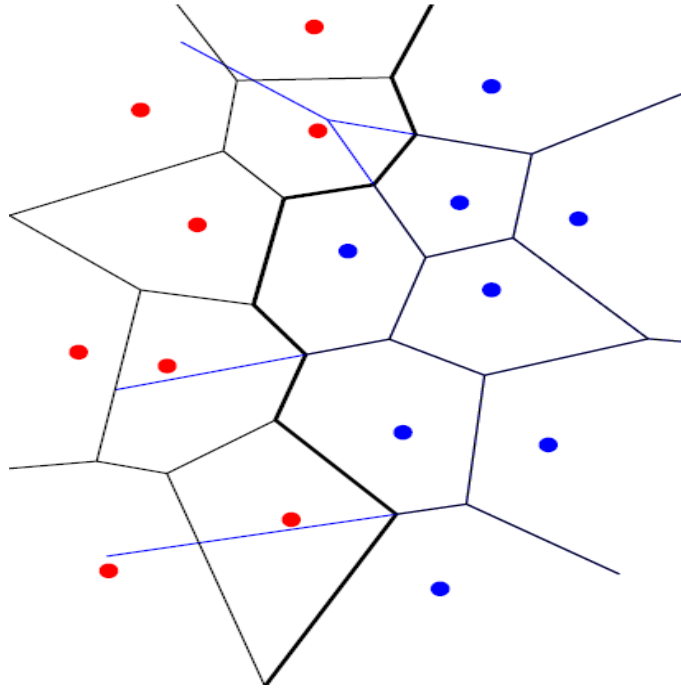


Figure 2.1 Divide and conquer approach

The bisector $b(R, B)$ is a y -monotone chain leaving the regions of the points $p_i \in R$ to its left and those of $p_j \in B$ to its right. Let R and B respectively be the regions of the plane located to the left and to the right of $b(R, B)$. Then $\text{Vor}(P)$ consists of $\text{Vor}(R) \cap \pi_R$, $\text{Vor}(B) \cap \pi_B$ and $b(R, B)$. Let e be an edge of $\text{Vor}(P)$. If e separates two points of R in $\text{Vor}(P)$, then it is (a portion of) the edge separating them in $\text{Vor}(R)$. Due to Observation 2, e cannot belong to π_B . If e separates two points of B , the case is analogous. If e separates one point of R from one of B , then $e \in b(R, B)$.

Algorithm 2.1 Divide And Conquer Algorithm

1. Sort the points of P by abscissa (only once) and vertically partition P into two subsets R and B , of approximately the same size.
2. Recursively compute $\text{Vor}(R)$ and $\text{Vor}(B)$.
3. Compute the separating chain.
4. Prune the portion of $\text{Vor}(R)$ lying to the right of the chain and the portion of $\text{Vor}(B)$ lying to its left.

2.1.2 Construction by Transformation

By transforming geometrical problems into easily understandable and solved problems is very helpful in solving the geometrical problems. Following in this way many attempts have been made to transform other well-known geometrical algorithms into the voronoi construction algorithm [2]. These include one dimensional reduction, Delaunay triangulation and other higher order dimensional embedding. The transformation process takes $O(n)$ time in most of the cases and hence construction efficiency depend on the construct from which it is derived.

2.1.3 Construction through Delaunay Triangulation

The planar voronoi diagram and Delaunay triangulation are dual of each other. Delaunay triangles correspond to voronoi vertices and Delaunay sites correspond to voronoi regions. Hence a voronoi diagram can be made from Delaunay triangulation in $O(n)$ time [6].

The graph G has a node for every Voronoi cell—equivalently, for every site—and it has an arc between two nodes if the corresponding cells share an edge. Note that this means that G has an arc for every edge of $\text{Vor}(P)$. In given figure, there is a one-to-one correspondence between the bounded faces of G and the vertices of $\text{Vor}(P)$.

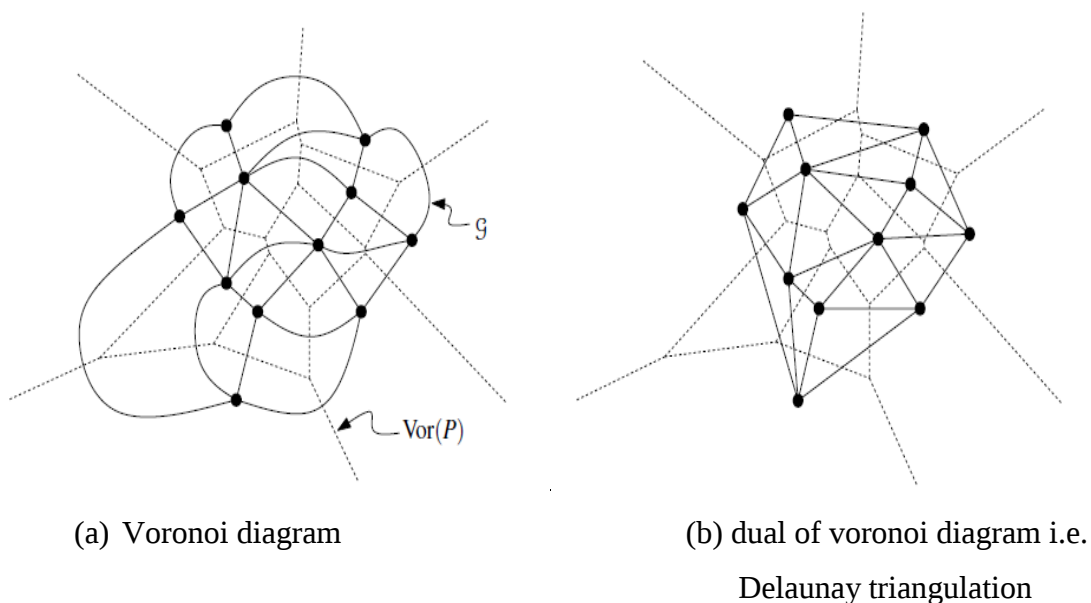


Figure 2.2 Voronoi and its dual Delaunay triangulation

2.1.4 Higher Dimensional Embedding

In geometrical sense, the convex hull is the dual of the voronoi diagram of given sites. The voronoi diagram in R^2 can be converted into convex hull in R^3 and vice versa of that in $O(n)$ time. Easy generalization to higher dimension is the main feature of embedding method. For determining convex hull of higher dimensions, many algorithms are known. Therefore an efficient method for computing worst case computation of d-dimensional voronoi diagram exists.

2.1.5 Incremental Construction

One of the method famous for its simplicity is the incremental method of constructing voronoi diagram in the plane in an incremental way in which the construction begin by initially few points and then remaining points are added one by one. When new site is entered the whole new voronoi diagram for all the sites including the site entered recently is constructed and displayed. The whole process of construction in this way can be divided into two major parts. In first part, the region of new site entered in old diagram is found. In second part, the boundary of new region of entered site is calculated edge by edge. This is done by calculating the edge between the newly entered site and the old sites that will share a common boundary with newly entered site [7].

This algorithm is much simple though its time complexity is $O(n^2)$ hence it cannot be used where speed matters. That resulted in further search for improving it. One such improvement made by researchers was that they tried to speed up the insertion up using the randomization [8]. As with other geometrical algorithms, numerical error comes up in the process of construction using incremental approach which is the major drawback of this algorithm. Some proposals were made to make the strategy of insertion robust against numerical error [9].

Algorithm 2.2 Incremental Algorithm

1. Starting with the Voronoi diagram of $\{p_1 \dots p_i\}$ add point p_{i+1}
2. Explore all candidate to find the step p_j ($1 \leq j \leq i$) closest to p_{i+1}
3. Compute its region
4. Build its boundary starting from bisector b_{i+1}
5. Prune the initial diagram.

6. While building the Voronoi region of p_{i+1} , UPDATE DCEL(Doubly Connected Edge List).

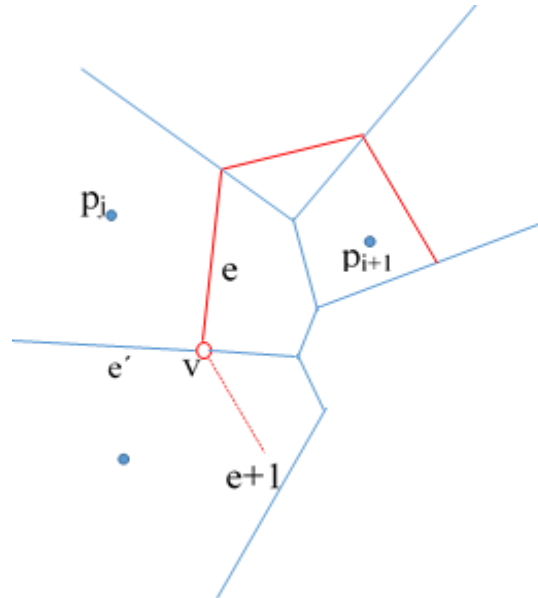


Figure 2.3 Incremental algorithm in progress

UPDATE DCEL

Each time an edge e , generated by p_{i+1} and p_j , intersects a preexistent edge, e' , a new vertex V is created and a new edge starts, $e + 1$. Then, these are the tasks to perform:

1. Assign $V_E(e) = V$, $e_N(e) = e'$, $f_L(e) = i + 1$, $f_R(e) = j$
2. Create $e+1$ and assign $V_B(e+1) = V$, $e_P(e + 1) = e$
3. Delete all edges of the region of p_j , that lie between $V_B(e)$ and $V_E(e)$ in clockwise order
4. Update $e(p_j) = e$
5. Create v with $e(v) = e$

Running time: Each step runs in $O(i)$ time, therefore the total running time of the algorithm is $O(n^2)$.

2.1.6 Dynamic Construction

As with many data structure, voronoi diagram can be made dynamic so that it can maintain set of sites which are added and deleted randomly. It is easy to handle the case of addition but for deletion suitable data structure is required to support deletion process. Attempts have been made to handle insertions and deletions of sites in $O(n)$ time with the help of voronoi tree that occupies $O(n \log n)$ space [10].

2.1.7 Parallel Construction

Parallelizing the algorithms in the field of computational geometry are not an easy task because of the sequential nature of many techniques used for it. Because the voronoi diagrams are of great importance, efforts have been made [11] to make it work in a parallel environment. Parallel algorithm for construction of voronoi diagram has been suggested by some researchers [12].

2.1.8 Plane-Sweep Construction

Another useful algorithm in the field of computational geometry to construct voronoi diagram is the plane sweep algorithm [13]. This works by decreasing the dimension of the problem as opposed to embedding method. The static problem of construction of voronoi diagram is converted to a dynamic problem of storing the cross section of voronoi diagram with straight line. This algorithm maintains a sweep line that goes from one side of the diagram to other sweeping through the sites of voronoi diagram. At any stage the swept portion of the diagram is complete and the remaining not swept area is incomplete and yet to be discovered by the sweep line.

Fortune [14] initially observed that updating the sweep line can be used with cost of $O(\log n)$ time when certain continuous deformation of the diagram is treated, and from this deformed diagram original diagram can be made in the time of order $O(n)$. The plane sweep method of constructing voronoi diagram is simple as well as efficient. It takes time of order $O(n \log n)$ and space of order $O(n)$ and hence it is one of best algorithms known to construct voronoi diagram.

A simple method to construct voronoi diagram is: for each site p_i , calculate the common intersection of the half-planes $h(p_i, p_j)$ with $j \neq i$, using the naïve algorithm. In this manner time of order $O(n \log n)$ is spent on every voronoi cell summing upto $O(n^2 \log n)$ time spent on construction of full voronoi diagram. Voronoi diagram has the complexity of the linear order. The plane sweep algorithm used in this thesis

work, also known as the Fortune's algorithm takes $O(n \log n)$ time to construct the voronoi diagram. The problem of sorting n numbers is reducible to the problem of computing the voronoi diagram in the plane. Hence any algorithm must take $\Omega(n \log n)$ time to compute the voronoi diagram in the worst case. Therefore the Fortune's algorithm is optimal.

2.2 Literature review of facility location problem

The competitive facility location problems has been investigated in many papers and been a subject of interest for many researchers. In most of the papers competitive location model for two competitors are given. Three companies that are in mutual competition to each other and intend to locate their facility on linear market [15]. It is known that Nash equilibrium solution does not exist for location problem of three or more competitive facilities.

The demands are continuously distributed on the market and facilities are located in some specific order of sequence A, B and C. Stackelberg equilibrium solution for three competitive facilities are considered. It considered the decision problems of three stages. In the first stage problem it consider the facility location for A so that it is optimal with respect to B and C. In the second stage problems it finds the optimal location of facility B with respect to facility C by using the information related to facility A. In final stage problem it finds the optimal location of the facility C by utilizing the information stored in the facilities A and B. This model has been represented as three stage decision problems.

Mobile facility location problem assigns each facility and client a start location in a metric graph. A destination node for each client and facility is to be found such that every client is sent to a node, same as that of destination of some facility. The total distance clients and facilities travels or by the maximum distance traveled by any client or facility determines the quality of a solution. The total movement of facilities and clients is minimized in [16] which generalize the classical k-median problem. [Demaine et al. in SODA 2007] introduced class of movement problems where it was observed a simple 2-approximation for the minimum maximum movement mobile facility location while an approximation for the minimum total movement variant and hardness results for both were left as open problems.

An 8- approximation algorithm for the minimum total movement mobile facility location problem main result here was main result. This problem generalizes the classical k-median problem preserving reduction. There cannot be better than a 2-approximation for the minimum maximum movement mobile facility location problem, unless $P = NP$; so the simple algorithm observed is essentially best possible.

Many location researchers have challenged the difficulties in multi-factor analysis of location decisions. The development of a novel geographic information system, based decision support system (GISDSS), is proposed by C. Jungthirapanich, and T. Pratheepthaweephon [17] for supporting high quality decision making in the facility location domain. A chromatic representation location model and vastly accepted location factors is incorporated by GISDSS. To manipulate data, and identify suitable sites, a geographic information system (GIS) is used with the location model. Input data is analyzed by the model and provides the best locations through the hue, saturation, and value (I-ISV) color model.

A unique color with its own chromatic representation is assigned to each location factor. The color saturation is varied, ranging from 0 as the most important to 1 as unimportant, to express the levels of importance of location factor. Variance in vertical value (V) depicts the scores for location alternatives, with 0 as the highest scores and 1 as the lowest. Thus with the combination of H, S, and V composite color can be visualized. The color displayed can also be stated quantitatively using the color equation. Thus recommended location can be effectively specified in both numerical and graphical formats.

To determine the location of the facility and the allocation of the demands of customers is the essence of all the facility location problems (under the condition of the minimum cost). An all-purpose bi-level simulated annealing algorithm (BSA) has been presented by Ren Peng [18] for the facility location problem, which is based on character and the idea of the standard simulated annealing algorithm. To solve the problem, the BSA is divided into two layers as inner layer and outer layer. For the decision of the facility location, outer algorithm is optimized and inner algorithm is optimized for the allocation of customer's demand under the given decision of the outer algorithm.

The hierarchical facility costs are a special case of the setting in which the facility cost is a more complex function of the set of clients, assigned to a single facility, and the algorithm, for the problem independent of the number of levels in the hierarchy tree and for the case of identical costs on all facilities, does not simply depend on their number. The bound is improved to 4.236 using scaling and accepting only sufficiently large improvements, it can be turned into a polynomial time $(4.236 + \epsilon)$ -approximation algorithm for the hierarchical facility location problem [19]. A facility cost, that is an arbitrary sub-modular function $\text{cost}(S)$ of the set of clients S assigned to the facility, defines a more general class of such problems.

With multiple transport alternatives, Yosuke Takano and others, [20] presented a modeling and optimization of facility location and distribution planning problems. The problem is to determine an optimal facility location with respect to management strategy, by using huge physical distribute on data. Two types of problems are considered. The first one is a facility location problem with transportation from depot to customer and a direct transportation from factory to customer simultaneously. Large size problem is solved efficiently by applying lagrangian relaxation. The second one is the competitive facility location problem with multiple competing companies. With this distribution profit's effectiveness is shown by collaborative decision making.

For the facility location optimization problems, which has earned extensive research interests, Maximal covering location problem (MCLP) is one of the well-known model. However the application of the traditional formulation of MCLP is limited by various practical requirements and effective approaches for large scale problems is made extremely difficult by the NP-hard characteristic. Li Xia and others [21] focused on a facility location problem motivated by a practical project of bank branching. The traditional MCLP formulation has been generalized as a mixed integer programming (MIP) with considerations of various costs and revenues, multi-type of facilities and flexible coverage functions. For large scale problems, a CPLEX -based hybrid nested partition algorithm has been developed and to deal with extremely large problems, heuristic-based extensions have been introduced. The formulation and algorithm are embedded into an asset called IFAO-SIMO. The effectiveness and efficiency of the approach is demonstrated by the numerical results.

Shuming Wang and others [22] have dealt with problems, under a hybrid uncertain environment, involving randomness and fuzziness. A two-stage fuzzy random facility location model, with recourse, is developed in which the demand and the cost are assumed to be fuzzy random variables. As in general, the fuzzy random parameters in the model can be regarded as continuous fuzzy random variables with infinite realizations, the computation of the recourse requires solving a large number of second-stage programming problems. Due to this fact, the recourse function cannot be calculated analytically, which implies that the model cannot benefit from the use of methods of classical mathematical programming. A technique of fuzzy random simulation is developed in order to solve the location problems of this nature. In the sequel, by combining the fuzzy random simulation, *i.e.* simplex algorithm and binary particle swarm optimization (BPSO), a hybrid algorithm is proposed for solving the two-stage fuzzy random facility location model.

Based on Plant Growth Simulation Algorithm (PGSA), Li tong with others [23] proposed a bionics algorithm to solve facility location problems. On comparing the calculating results of PGSA with Genetic Algorithm (GA) for distribution center location problem, it is found that PGSA is better than GA in term of accuracy. By selecting 50 customers randomly, it also solve Weber location problem with different facility numbers. With respect to other heuristic algorithms, PGSA can find global optimal solutions. Meanwhile, according to the different facility numbers, it combines global and local optimal solutions and set up optimal facility location arrangement as a whole. The algorithm discussed here shows its accuracy, astringency and generalization. Solving location problems is an actual application of PGSA.

Timeliness is one of the most important objectives as it reflects the quality of emergency rescue. To increase the number of service facility available is the most obvious way for providing timeliness. Unfortunately, due to capital constraints, increasing the number of facility is generally impossible. In such a case, the strategy for emergency facility location becomes an important issue. YU Dejian with others [24] discussed the facility location strategy in emergency management, combining subjective judgment and objective analysis, and proposed an emergency system location model which is based on weighted grey target strategy theory. Finally, an application example verified that the method is effective one for solving the emergency facility location issue.

The Multiple Facility Location Problem (MFLP) is to locate certain facilities so as to serve optimally a given set of customers, whose requirements and locations are known. When facility locations have to be selected from a given set of locations, the corresponding location problem becomes a Discrete Multiple Facility Location Problem (DMFLP). In this study Davood Shishebori with others [25] considered a special case of DMFLP where multiple facilities that are of different type are placed (location decision) and assigned customers to these facilities (allocation or assignment). The case will be discussed based on interactions (with and without) among new facilities. Then proposed are new heuristic solution methods and branch-and-bound algorithms. Computational results on randomly generated data, in comparison with optimal solutions, indicates that the new methods are, both, accurate and efficient.

Bin Yi+ and Rongheng Li [26] considered one kind of uncapacitated facility location problem which is termed as k-product uncapacitated facility location problem with no-fixed costs (k-PUFLPN). The problem can be defined as follows: There is a set of demand points, where clients are located and a set of potential sites, where facilities of unlimited capacities can be set up. K different kinds of products are there. Each client needs to be supplied with k different kinds of products by a set of k different facilities and each facility can be set up to supply only a distinct product, with no fixed cost. A non-negative cost of shipping goods is there in between each pair of locations. These costs are assumed to be symmetric and also satisfy the triangle inequality. A set of facilities, which are to be opened, and their designated products is to be selected and has to find an assignment for each client within a set of k facilities to minimize the sum of the shipping costs. In this paper, an approximation algorithm was proposed with a performance guarantee of $(3/2)^{k-1}$ for the k-PUFLPN.

The continuous growth of wireless sensor networks demands new methods and approaches to efficiently manage and service them. Eliaz Velazquez and others [27] presents an approximate solution to the facility location problem for sensor network maintenance, which is based on static sensors and mobile facilities. To increase the network lifetime by recharging or redeploying sensors with the help of mobile multi-purpose maintenance facilities is the main objective. Problem is variant of the facility location problem (FLP). In this case, a suitable deployment of the facilities is to be done in which their workload is balanced and the movement of the facilities, in their

area, is minimized. It should be accomplished keeping the number of sensor communications at minimum. While finding the optimal placement of the maintenance facilities is a NP-hard problem, this work [27] showed a simple and efficient solution, totally distributed and localized, which starts with a balanced deployment, progresses towards final partition of remarkable quality. Such a final partition satisfies the load balancing requirement and minimizes the facility travel time. The experimental analysis of such solution shows that sensor message cost remains low as the size of the network increases. The experiments also show a load distribution which is similar and sometimes better than centralized deployment solutions.

Environmental regulations are forcing companies to comply with environmental policies so as to control carbon emission. It is required for companies to green their supply chains. One way to do this is extending the supply chain to collection and recovery of products in closed loop configuration. Profitable reverse logistics to restore the recovered product can be used so as to resell it in primary or secondary market. Ali Diabat and others [28] have introduced a multi-echelon multi-commodity facility location problem with a trading price of carbon emissions and a cost of procurement. If carbon cap is higher than the total emission then company gains but if carbon cap is less than the total emission then company might incur cost.

3.1 Gap analysis and related work

Facility location problem or location analysis or problem of k-center is a branch of computational geometry and operations research that deals with location of facilities so as to optimize specific requirement in the problem. It convert real life problem into mathematical problem and then modeling it to find the solution of the problem using many facility location algorithms. The target of problem solving can be minimizing transportation costs, placing harmful substance or radio-active substances at farthest location or locate facility in a competitive environment. Initially used for locating facilities, this field now has been expanded covering many advance fields like data clustering, classification, databases, unsupervised learning, data mining, spatial range query and spatial query integrity [29].

In its basic form, the facility location problem contains a set of facilities F from which a subset of facilities, F_i , are to be placed so as to satisfy demands of set of demand points C . The goal here is to locate the facilities in such a way so as to minimize the distance from each facility to demand points. Sum of costs of opening the facilities should be minimized.

The Facility Location problem when considered on general graphs is an NP-hard problem to be solved optimally, by reduction from the Set Cover problem or any other problem of this type. Many approximation algorithms have been proposed for solving the facility location (FP) problem and other variants of it because of the usefulness of the facility location problem in real life and many other fields.

3.1.1 Nearest neighbor search

In problem of nearest neighbor searching, a data points set of n points is given in a metric space, Q . The goal is to preprocess these given data points so that, given any query point $q \in Q$, a data point nearest to q can be found quickly. This is also known as the post office problem or the closest point problem in computational geometry.

3.1.2 Minisum facility location

It is a simple facility location problem also known as Fermat-Weber problem. In this problem it is required to find a single point from which the sum of a given set of query points is minimum. In other words it can be stated as in Fermat-Weber problem a single point in space is located within a set of points with an optimization criteria of minimizing the sum of the distances of the points from other points. Some variations of above stated problem exists in which it is desired to place multiple facilities or there are more complex optimization requirements or constraints on the locations of facilities are there.

3.1.3 Minimax facility location

As it is obvious from the name of the problem, in this category of problems it is required to find or locate a point in the space so as to minimize the maximum distance between this point and sites where distance means the minimum distance between point and any one of the sites given.

3.1.4 Maxmin facility location

The maxmin facility location problem is the reverse of the minimax facility location problem. In this problem it is required to find or locate a point which maximizes the minimum distance of the point from the sites where distance means the maximum distance from point to any one of the sites given.

3.2 Problem statement

Given m facilities and n customers in a city, find the facility which is located at such a place so that the sum of distances from a set of customers is minimum. Also the maximum difference in distance for any two customers should be minimized to best possible solution preserving former criteria.

3.3 Problem formulation

Let $F\{f_1, f_2, \dots, f_m\}$ is a set of facilities located at different places and $C\{c_1, c_2, \dots, c_n\}$ is a set of n customer residing at different location.

$C_i \subseteq C$, for $i=1, 2, \dots, 2^n$

$\text{dist}(x, y)$ = distance between x and y

(i) Maximum difference, $MD(f_j, C_i) = \text{MAX}[\text{dist}(f_j, c_p)] - \text{MIN}[\text{dist}(f_j, c_p)] \quad \forall c_p \in C_i$

(ii) Aggregate Distance, $AD(f_j, C_i) = \sum \text{dist}(f_j, c_k), \forall c_k \in C_i$

To find a facility f_i that minimizes the maximum difference and aggregate distance i.e. which minimizes the functions (i) and (ii) both.

The problem of facility location is old enough and is solved by many researchers in many ways. Some of the major problem types of facility location problem are nearest neighbor search, Minisum facility location, Minimax facility location and Maxmin facility location.

- Nearest neighbor search mainly deals with single query point searching for nearest facility.
- Minisum or Fermat-Weber problem deals with only minimizing the aggregate sum of distances of all the customer from the facility.
- Minimax focusses only on minimizing maximum distance of any customer from the facility.
- Maxmin focused only on maximizing minimum distance of any customer from the facility.

In real life many times there are situations where an optimal solution is required which is the combination of solutions to each of the above problems. These problems deal with only one or two criteria. Hence a method is required which gives the optimal solution which has all of the above conditions satisfied. The algorithm proposed in this thesis tries to find out the approximately optimal solution in less time.

4.1 Preliminary of Fortune's algorithm

4.1.1 Background of Fortune's algorithm

In plane sweep line algorithm, a horizontal line goes from one side to opposite side sweeping through all the sites of voronoi diagram. During this process, the information of structure computed is maintained in data structures. When sweep line sweeps through the input voronoi sites, the information regarding the intersection of this line with the input voronoi sites area. During sweep, most of the time information stored remains same except during some special events known as site events.

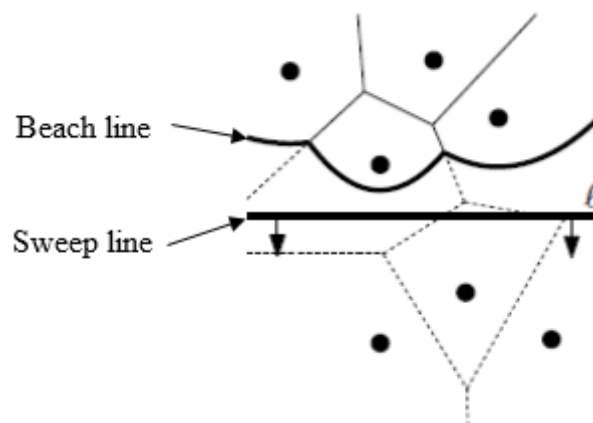


Figure 4.1 Beach line and sweep line

This technique of plane sweep algorithm can be applied to construction of voronoi diagram of a given set of site points, $P = \{p_1, p_2, \dots, p_n\}$ in a given plane. In this method a sweep line ℓ keeps sweeping through the sites in the given set of sites from top to bottom (or bottom to top). The important thing here is that it is required to store the information of intersection of sweep line with the area of given sites as the sweep line moves downward. But this is not an easy task because the structure above the sweep line does not depend on the sites above sweep line only but also on the sites below the sweep line. In other words, when sweep line meets a topmost vertex of any voronoi cell, it still doesn't have the information about the site of that cell. So it doesn't have all the necessary information required to compute the voronoi vertices of the cells

whose vertex is about to come and site is below the sweep line. Hence plane sweep method is needed to be applied in a slightly different manner.

Instead of keeping the information of intersection of the sweep line with the sites area, information of the structure above sweep line that does not change further on moving of sweep line is stored i.e. part of the voronoi diagram above sweep line that is not affected by the sites below the sweep line is stored because it is static will not change till the end of computation of voronoi diagram of given sites.



Figure 4.2 Beach line is x-monotone

If area above sweep line ℓ is denoted by ℓ^+ it is required to find out which part of voronoi diagram in area ℓ^+ will not change further i.e. the points in area ℓ^+ for which it is known about their nearest site. As it is obvious that the distance of any point q in ℓ^+ from any site below the sweep line ℓ will be more than the distance of q from sweep line ℓ itself. Hence all points q whose distance to its nearest site in area ℓ^+ is less than or equal to its distance from sweep line ℓ , will lie in the cell of that nearest site and hence area made of all such points will not change as the sweep line moves. The area of such points is bounded by the parabola made of that nearest site as the focus and the moving sweep line as its directrix. Hence all such points are bounded by their respective parabola boundaries. The continuous arc formed by the combination of lowest part of these parabolas is called as the beach line. Because

beach line is formed by the lowest parts of all parabolas above the sweep line and is continuous therefore it is x-monotone i.e. any vertical line cuts it at only one point.

It is observed that the one parabola can contribute to the beach line at more than one place. A breakpoint is a point of intersection of two parabolas or the point where one parabolic curve changes into other new curve. These breakpoints between the parabolic arcs define the edge of the voronoi diagram. These breakpoints traces out edges as the beach line moves.

Therefore beach line is maintained to construct voronoi diagram instead of storing the intersection of the sites area with the sweep line. The beach line is not stored explicitly as it keeps changing. It changes when any event either site event or circle event happens. In the site event a new arc appears on the beach line and in the circle event the arc disappear from the beach line and shrinks to a point that will define one of the vertex of the voronoi diagram.

When sweep line reaches a new site in the unexplored site area, a degenerate parabola whose vertex and focus is the site itself and lies on the sweep line as its directrix. In other words, the width of the parabola at that time is nil. Hence a vertical line appears at that moment that connects the new site with the beach line above it vertically. As the sweep line sweeps towards down, the new parabola continue to widen. Now the new parabolic arc becomes the part of the old beach line. This process, when a new site is encountered is known as a site event and it is described in the figure below.

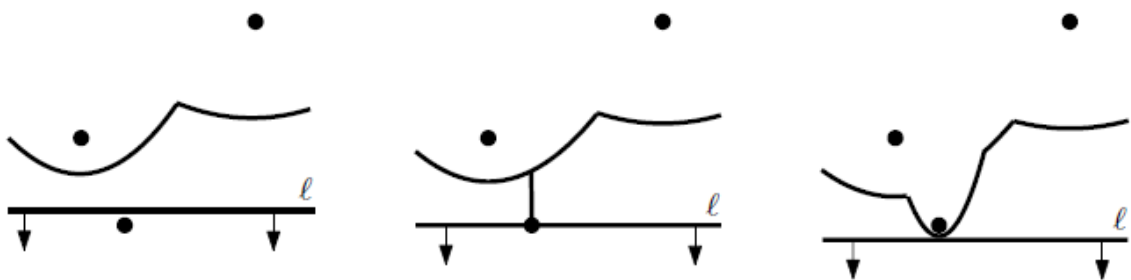


Figure 4.3 stages of a site event

At a site event, when the newly appeared parabola starts widening, it intersects old parabola at two points both acting as breakpoint of new beach line. These two breakpoints starts tracing out an edge of the voronoi diagram. At first this edge is not

connected to the part of the voronoi diagram constructed but later on this edge meets another edge hence creating a vertex of the voronoi diagram and becomes the part of already constructed voronoi diagram.

Whole of the above explanation can be summed up in the following points as the properties of Fortune's algorithm preliminaries:

- The beach line is monotonic in nature over x axis i.e. any vertical line can intersect it at only one point.
- Any new arc can appear on beach line through site event only
- Any arc on beach line can disappear from it through circle event only.
- Every vertex is detected by a circle event

4.1.2 Data structures for Fortune's algorithm

(a) Doubly connected edge list (D): it is required to store the subdivisions of voronoi diagram computed so far.

(b) Balance binary search tree (T): it is required to store beach line current status. Any leaf stores arc in the form of site forming it and internal nodes represent breakpoint stored in the form of tuple of sites $\langle p_i, p_j \rangle$.

(c) Priority queue (Q): it is required to store event queue where priority is decided by the y coordinates of the point.

Following is the algorithm to compute voronoi diagram using Fortune's algorithm as explained by Mark de Berg [1].

Algorithm 4.1 Voronoi diagram algorithm

VORONOIDIAGRAM(P)

Input. A set $P := \{p_1, \dots, p_n\}$ of point sites in the plane.

Output. The Voronoi diagram $\text{Vor}(P)$ given inside a bounding box in a doubly connected edge list D .

1. Initialize the event queue Q with all site events, initialize an empty status structure T and an empty doubly-connected edge list D .
2. while Q is not empty

3. do Remove the event with largest y -coordinate from Q .
4. if the event is a site event, occurring at site p_i
5. then HANDLESITEEVENT(p_i)
6. else HANDLECIRCLEEVENT(γ), where γ is the leaf of T representing the arc that will disappear
7. The internal nodes still present in T correspond to the half-infinite edges of the Voronoi diagram. Compute a bounding box that contains all vertices of the Voronoi diagram in its interior, and attach the half-infinite edges to the bounding box by updating the doubly-connected edge list appropriately.
8. Traverse the half-edges of the doubly-connected edge list to add the cell records and the pointers to and from them.

The procedures to handle the events are defined as follows Mark de Berg [1].

Algorithm 4.2 Handle site event algorithm

HANDLESITEEVENT(p_i)

1. If T is empty, insert p_i into it (so that T consists of a single leaf storing p_i) and return. Otherwise, continue with steps 2 - 5.
2. Search in T for the arc α vertically above p_i . If the leaf representing α has a pointer to a circle event in Q , then this circle event is a false alarm and it must be deleted from Q .
3. Replace the leaf of T that represents α with a sub-tree having three leaves. The middle leaf stores the new site p_i and the other two leaves store the site p_j that was originally stored with α . Store the tuples $\langle p_j, p_i \rangle$ and $\langle p_i, p_j \rangle$ representing the new breakpoints at the two new internal nodes. Perform rebalancing operations on T if necessary.
4. Create new half-edge records in the Voronoi diagram structure for the edge separating $V(p_i)$ and $V(p_j)$, which will be traced out by the two new breakpoints.
5. Check the triple of consecutive arcs where the new arc for p_i is the left arc to see if the breakpoints converge. If so, insert the circle event into Q and add pointers between the node in T and the node in Q . Do the same for the triple where the new arc is the right arc.

Algorithm 4.3 Handle circle event algorithm

HANDLECIRCLEEVENT(γ)

1. Delete the leaf γ that represents the disappearing arc α from T. Update the tuples representing the breakpoints at the internal nodes. Perform rebalancing operations on T if necessary. Delete all circle events involving α from Q; these can be found using the pointers from the predecessor and the successor of γ in T. (The circle event where α is the middle arc is currently being handled, and has already been deleted from Q.)
2. Add the center of the circle causing the event as a vertex record to the doubly-connected edge list D storing the Voronoi diagram under construction. Create two half-edge records corresponding to the new breakpoint of the beach line. Set the pointers between them appropriately. Attach the three new records to the half-edge records that end at the vertex.
3. Check the new triple of consecutive arcs that has the former left neighbor of α as its middle arc to see if the two breakpoints of the triple converge. If so, insert the corresponding circle event into Q. and set pointers between the new circle event in Q and the corresponding leaf of T. Do the same for the triple where the former right neighbor is the middle arc.

All the three algorithms above have been taken from Computational Geometry: algorithms and applications by Mark de Berg [1].

Table 4.1 Comparison of various algorithms of voronoi construction

Algorithm	Time complexity	Space complexity
Naïve algorithm	$O(n^2 \log n)$	$O(n^2)$
Incremental algorithm	$O(n^2)$	$O(n^2)$
Divide and conquer	$O(n \log n)$	$O(n^2)$
Fortune's algorithm	$O(n \log n)$	$O(n)$

4.2 Proposed algorithm

4.2.1 Algorithm for nearest facility location for multiple customers

The approach of finding the optimal facility is divided into two algorithms. First finding the voronoi diagram of facilities using any voronoi diagram construction algorithm. In this work, Fortune's algorithm is used to calculate the voronoi diagram. Second computing the location of optimal facility by the proposed algorithm taking voronoi diagram of facilities $Vor(F)$ and set of locations of customers C as input. The second algorithm uses the result of first algorithm to give output as a facility at optimal distances from all customers.

Algorithm 4.4 nearest facility location algorithm

NEARFACILOC($Vor(F)$, C)

Input: voronoi diagram of facilities $Vor(F)$, set of customer locations, C

Output: facility f_o optimal for all customers

1. Give as input voronoi diagram, $Vor(F)$ of facilities and locations of customers C .
2. Create voronoi diagram for customers at different location by using $VORONOIDIAGRAM(C)$.
3. Find all the vertices of this voronoi diagram created in step 2.
4. Find the coordinates of these vertices.
5. Calculate the mean, M , of these vertices.
6. Locate this mean in the original voronoi diagram of facilities.
7. Find in which region or cell this mean, M , lies.
8. Find the facility (site) of the above selected region or cell.
9. Output the facility, f_o found in above step as facility at optimal distances from all customers.

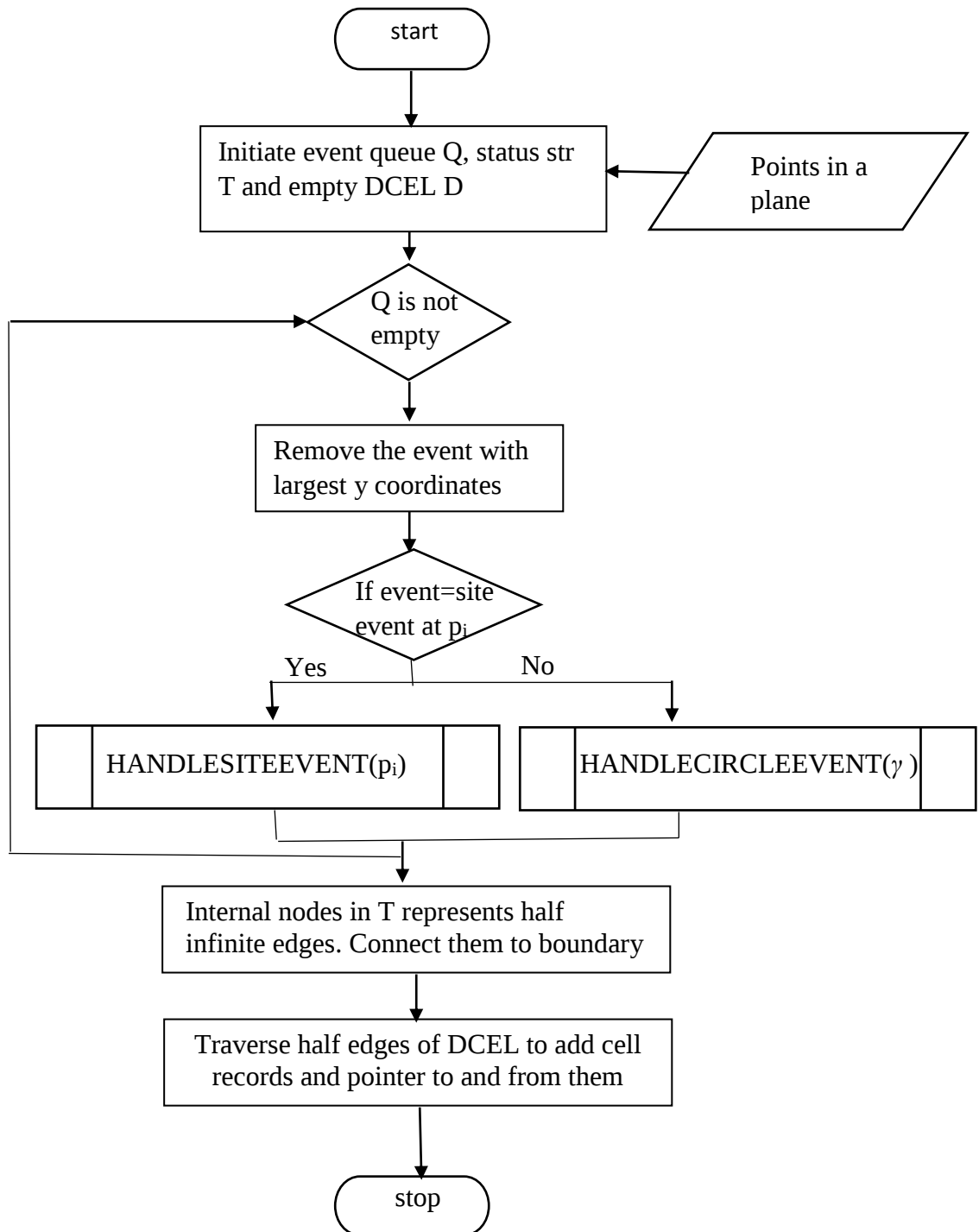


Figure 4.4 Flowchart of voronoi diagram construction using Fortune's plane sweep algorithm

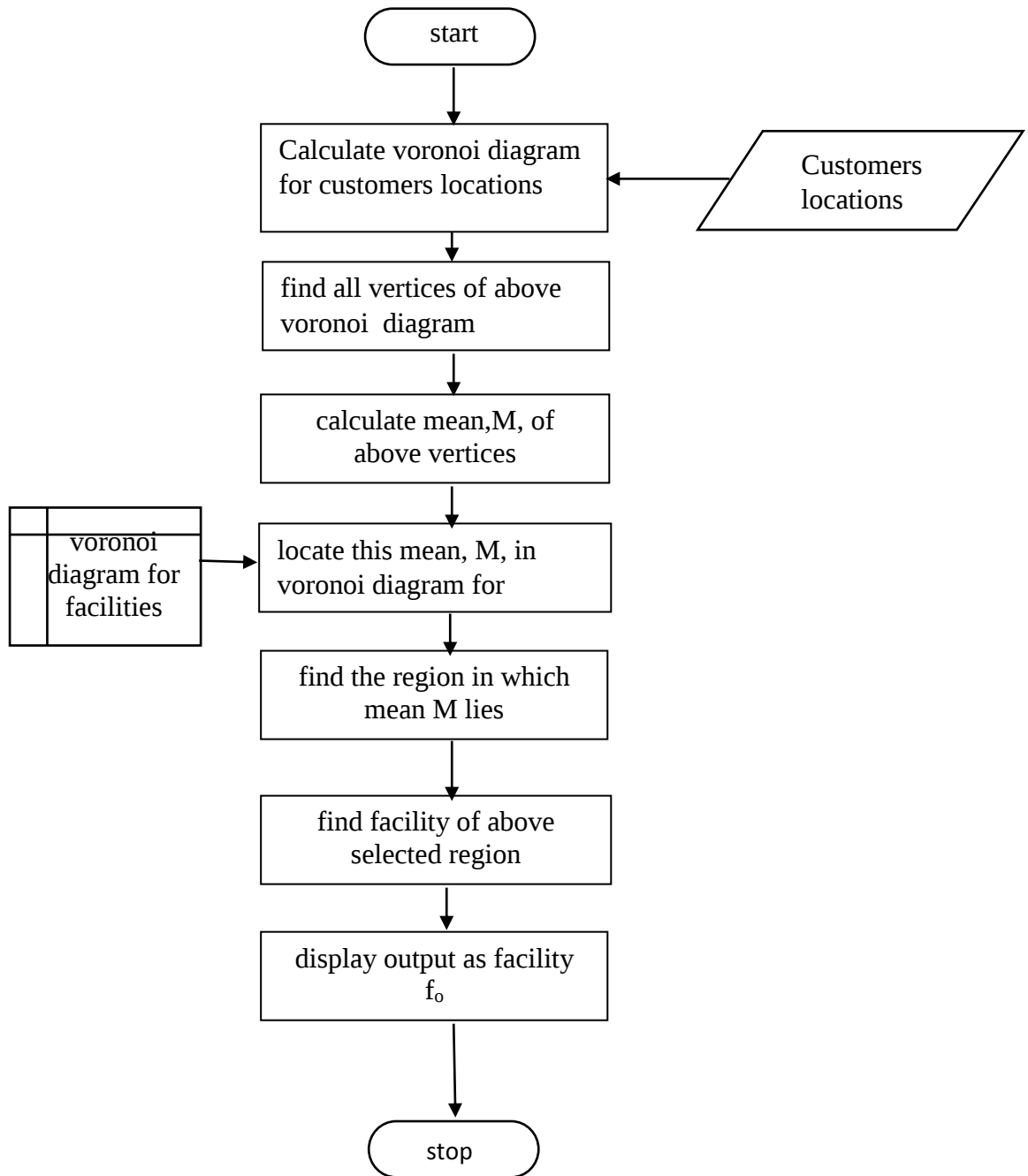


Figure 4.5 Flowchart of proposed algorithm

4.2.2 Explanation of working of proposed algorithm

The proposed algorithm takes voronoi diagram as computed by using Fortune's sweep line algorithm as one of the input, the customers location as being the other input of this algorithm.

First of all, by treating customers location as sites, voronoi diagram of customers location is computed using the Fortune's sweep line algorithm. The voronoi diagram is calculated in optimal time of $O(n \log n)$ which is the time complexity of Fortune's algorithm. Now in this computed voronoi diagram of customers locations, all the vertices are located. By using these vertices, average or mean of vertices is found. After calculating mean of new voronoi diagram of customers, the old voronoi diagram of facilities is required. The old voronoi diagram is stored in main memory and can be used as and when required. Then the calculated mean of new voronoi diagram is located in old voronoi diagram. When located, the region of old voronoi diagram in which mean lies, is found. Then the facility as a site of this selected region is found. This facility is the required output.

This facility will minimize the aggregate distance i.e. the sum of the distances of all the customers from this facility. Also this facility selected will be the one which tries to minimize the maximum difference i.e. the difference of the distance between the maximum distance and the minimum distance, by always preserving the first minimization criteria at priority (minimizing aggregate distance).

As the facilities are discrete objects, it is not always possible to have a site that gives minimum for both the optimizing functions. Hence it always give the minimum aggregate distance with best possible solution for minimum of maximum difference in combination with first optimizing criteria.

The time complexity of the proposed algorithm is same as time complexity of Fortune's algorithm because it uses Fortune's algorithm as the base algorithm. It spent constant amount of time on computing the mean. Hence overall complexity of proposed method is $O(n \log n)$.

Chapter 5

Testing and Results

5.1 Test case 1

The proposed algorithm was tested by running many test using different values. Its step by step process is given here. First of all, the different facilities (T) were distributed in space. In real world these facilities may represent anything like theatre or hospital in a city. The exact locations of facilities are shown in terms of coordinates. These locations are fixed for a scenario i.e. the facilities are distributed in space are static in nature. They are not moving in space.

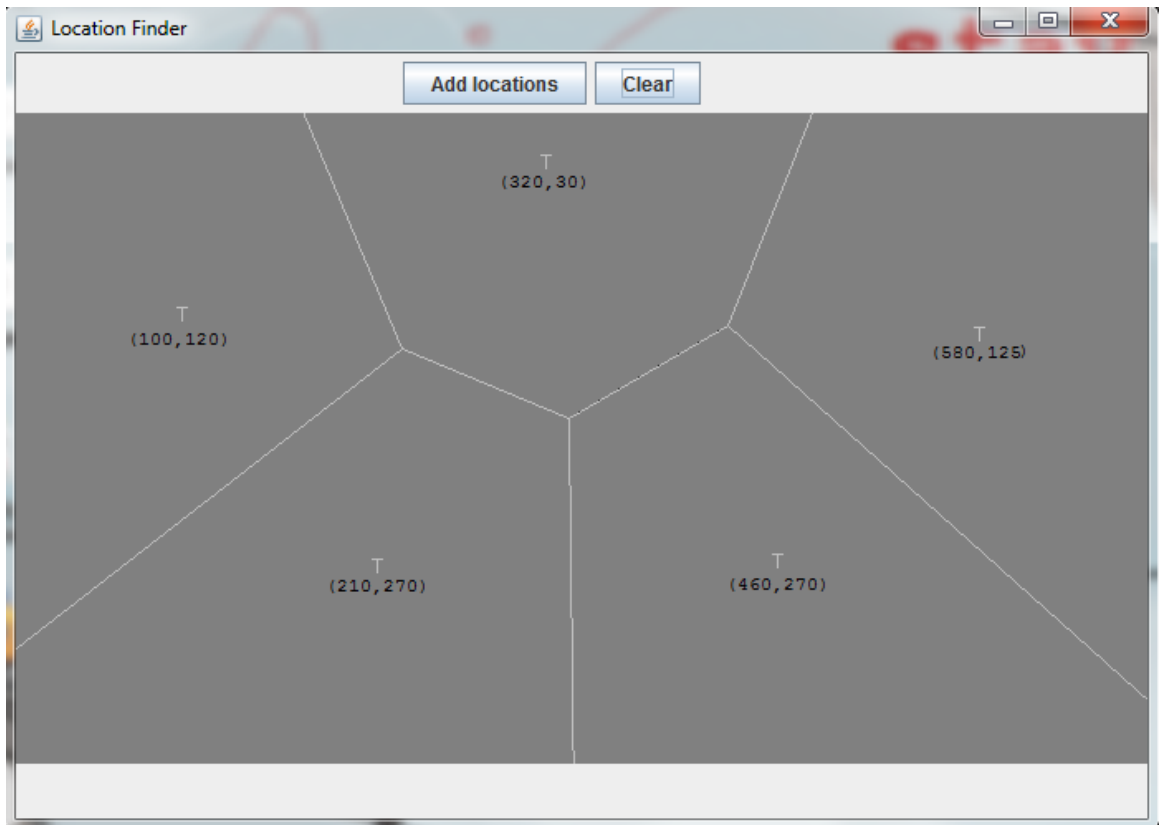


Figure 5.1 Facilities located for test case1

These facilities are distributed by clicking on the screen at appropriate locations. When these facilities were located on the screen, the voronoi diagram of these facilities was created simultaneously using the voronoi construction algorithm. Here fortune's sweep line algorithm for construction of voronoi diagram has been used.

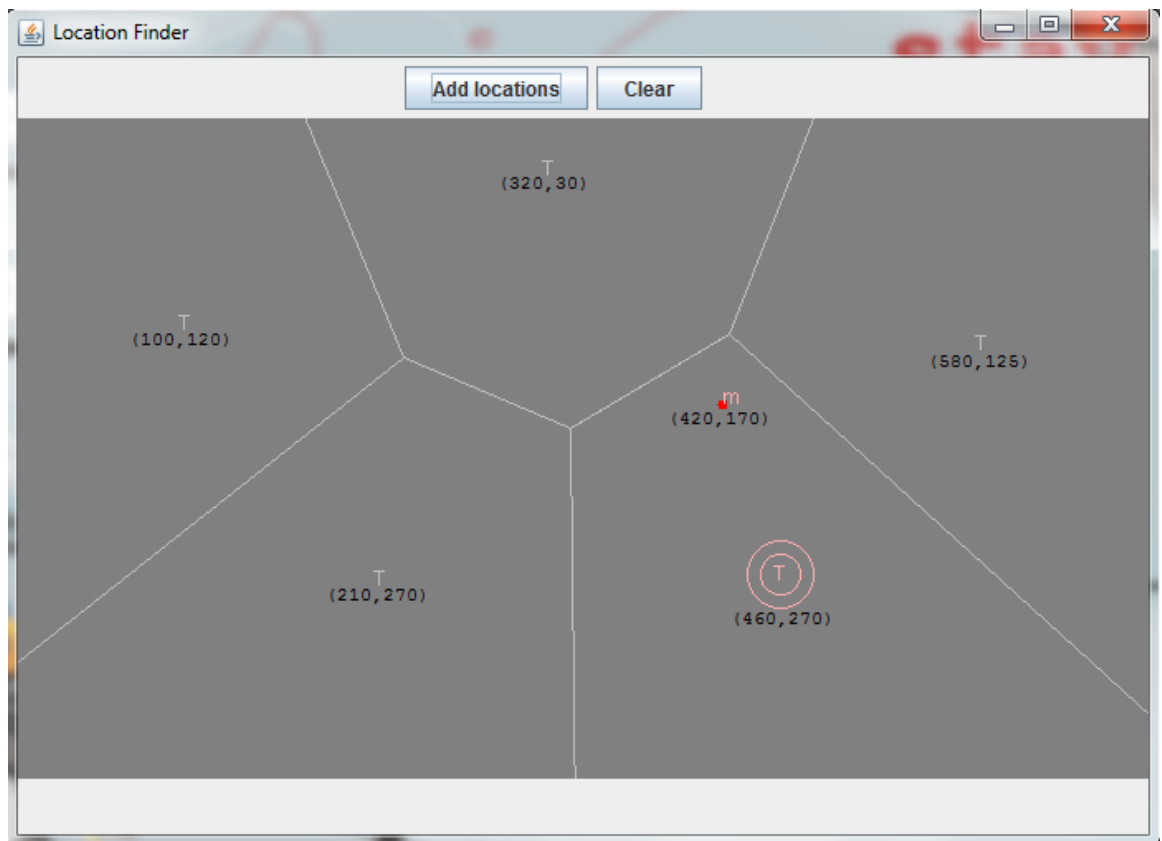


Figure 5.2 After one customer added in test case 1

When all sites or facilities are located, “add location” button is clicked to add the locations of the customers. The facilities in real world may represent any objective location like hospital, chemist, mall or theatre located in a big city. One such partition will be computed for any type of facilities input. For one type of facilities input, multiple queries can be done on that resulted voronoi diagram of the facilities located. Different customers may live at different parts of the city.

One such customer is added in the figure shown above represented with the red dot. When one input as a customer is given, the proposed method just calculates the mean, m . In this case mean, m , and the customer’s location will be the same. Now it checks for in which region this mean m lies. When found the region it shows the site of the corresponding region as a result.

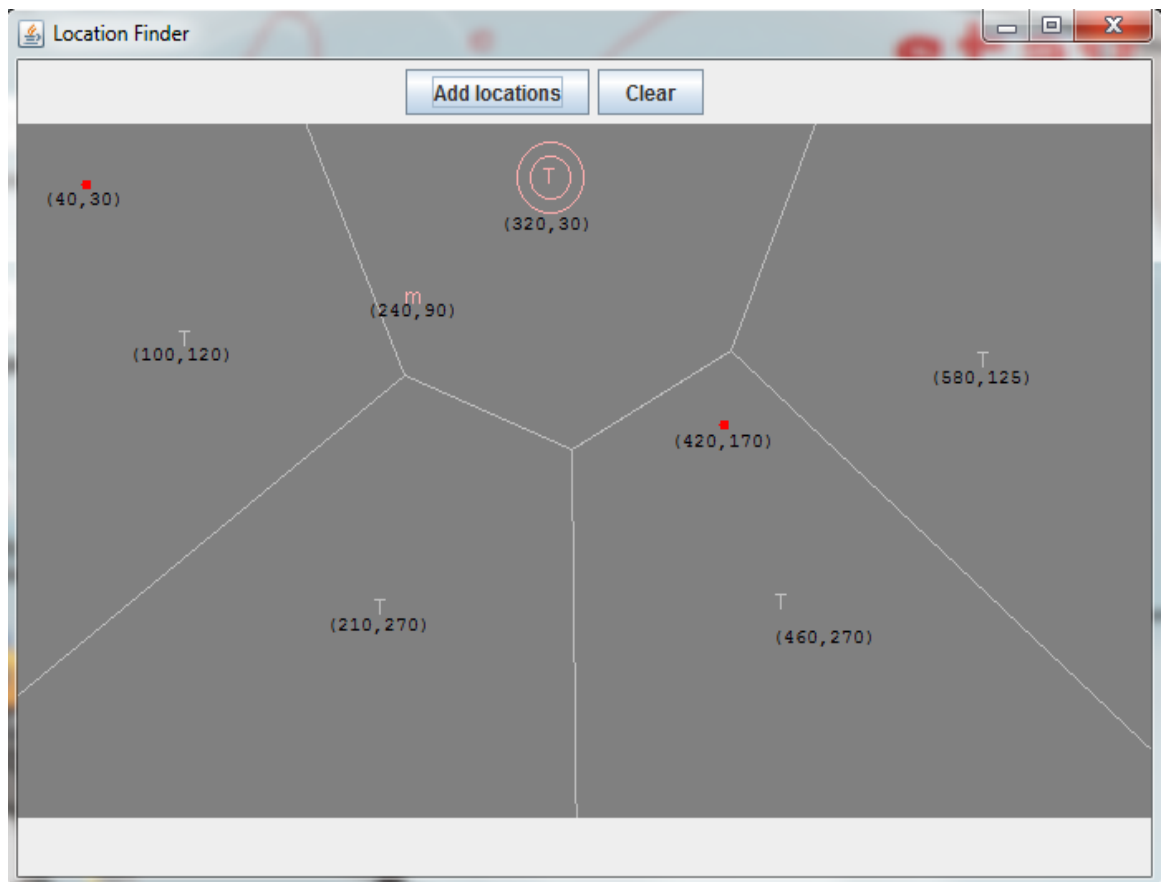


Figure 5.3 After two customer added in test case 1

Location of second customer is added by clicking at proper place. When location of this customer is added the corresponding voronoi mean, m , is calculated by the proposed method. Then it calculates the region in which this voronoi mean, m , lies. When found the region in which the calculated voronoi mean, m , lies, it displays the site corresponding to that region as doubly co-centric circles. For two customers, the mean always lies at the mid of the line joining the two of them. It is also valid form the point of view of real world scenario, when two persons want to meet at a particular location they choose the location which is approximately equidistant from both. Hence it works correctly for two customers.

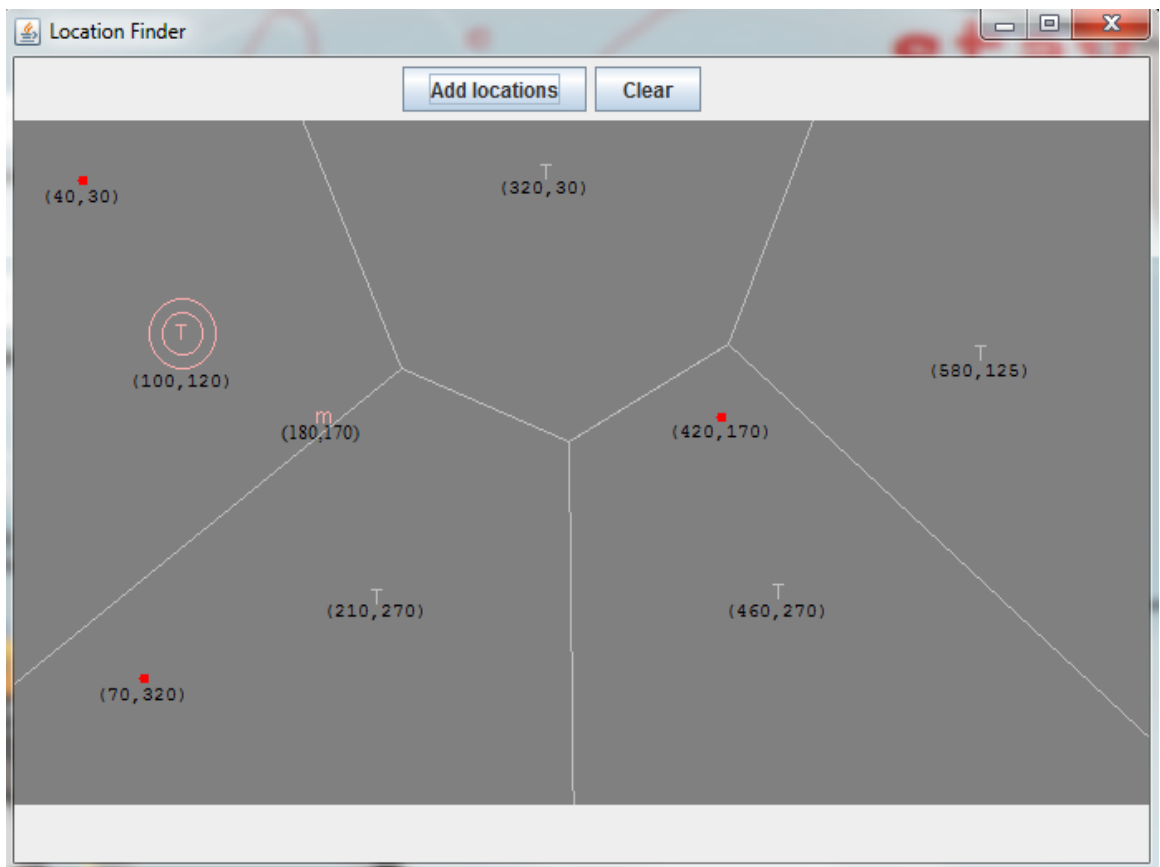


Figure 5.4 After three customers added to test case 1

Location of third customer is added by clicking at proper place. When the location of this customer is added, the corresponding voronoi mean, m , is again calculated by the proposed method. Now the mean is shifted from the previous location towards the new region that will be the one containing the desired optimal facility site. Then it calculates the region in which this voronoi mean, m , lies. When found the region in which the calculated voronoi mean, m , lies, it displays the site (facility) corresponding to that region as doubly co-centric circles. In the above situation for three customers, the mean lies at the circumcenter of the triangle formed by three customers as the corners of the triangle. From the perspective of real world three persons would like to meet at a location approximately equidistant from three of them. The circumcenter of a triangle lies at equidistant from all three points of the triangle, hence the mean satisfy what is expected from the method

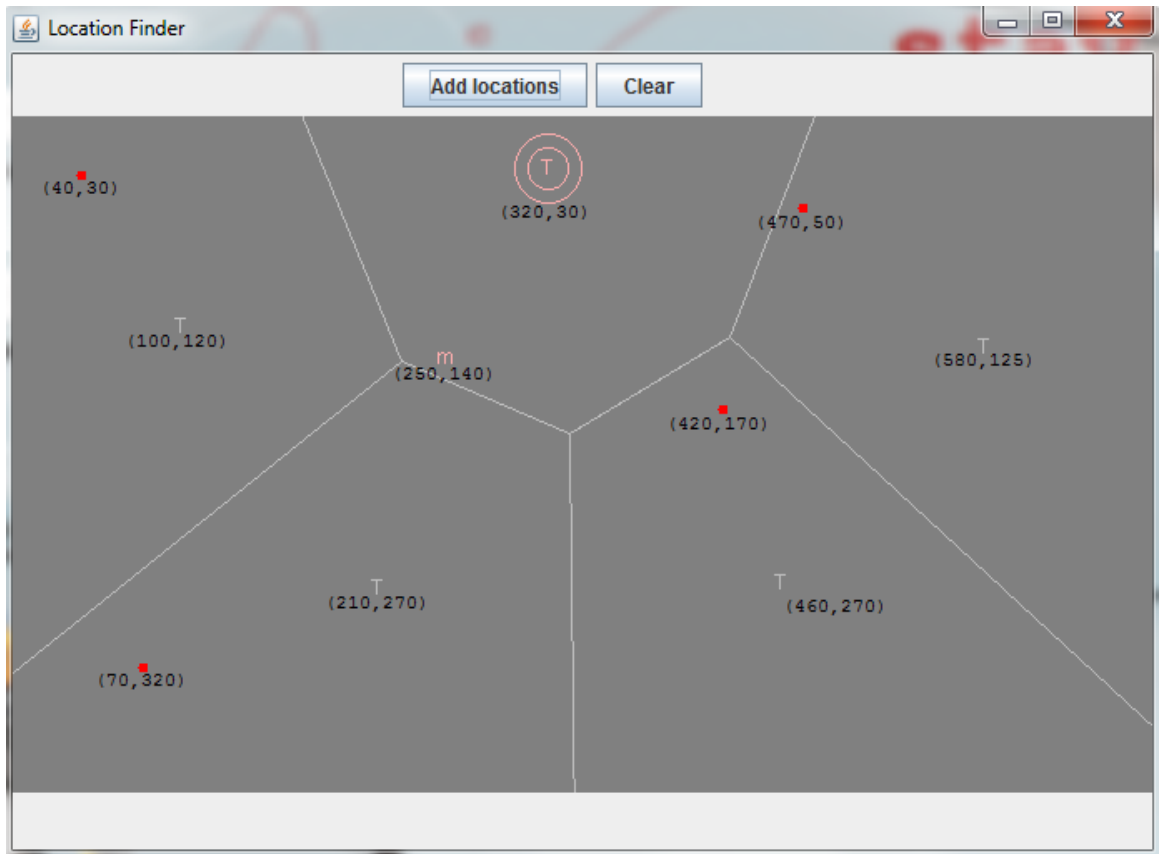


Figure 5.5 After four customer added to test case 1

Location of fourth customer is added by clicking at proper place. When the location of this customer is added, the corresponding voronoi mean, m , is again calculated by the proposed method. Now the mean is shifted from the previous location towards the new region that will be the one containing the desired optimal facility site. Then it calculates the region in which this voronoi mean, m , lies. When found the region in which the calculated voronoi mean, m , lies, it displays the site (facility) corresponding to that region as doubly co-centric circles.

In this way any number of customers can be added to this system to find the facility at an optimal distance from every customer. It works dynamically as any customer can be added at any time and the resultant diagram is shown quickly.

5.1.1 Verification for test case 1

Table 5.1 Aggregate Distance for test case 1

Facility	Distance of Customer1 (40,30)	Distance of Customer 2 (70,320)	Distance of Customer 3 (420,170)	Distance of customer 4 (470,50)	Aggregate Distance (AD)
T1(100,120)	108.166	202.238	323.882	376.563	1010.849
T2(210,270)	294.109	148.661	232.594	340.588	1015.952
T3(320,30)	280	382.884	172.046	151.328	986.258
T4(460,270)	483.736	393.192	107.703	220.227	1204.858
T5(580,125)	548.293	546.008	166.208	133.135	1393.644

Table 5.2 Maximum difference for test case 1

Facility	Distance of Customer1 (40,30)	Distance of Customer 2 (70,320)	Distance of Customer 3 (420,170)	Distance of customer 4 (470,50)	Max distance	Min distance	Max Diff (MD)
T1 (100,120)	108.166	202.238	323.882	376.563	376.563	108.166	268.397
T2 (210,270)	294.109	148.661	232.594	340.588	340.588	148.661	191.927
T3 (320,30)	280	382.884	172.046	151.328	382.884	151.328	231.556
T4 (460,270)	483.736	393.192	107.703	220.227	483.736	107.703	376.033
T5 (580,125)	548.293	546.008	166.208	133.135	548.293	133.135	415.158

5.1.2 Result of test case 1

Facility T3(320,30) has minimum aggregate distance from all customers as shown in table 5.1. Also it minimized the maximum difference up to second best possible result. Hence proposed algorithm works fine here.

5.2 Test case 2

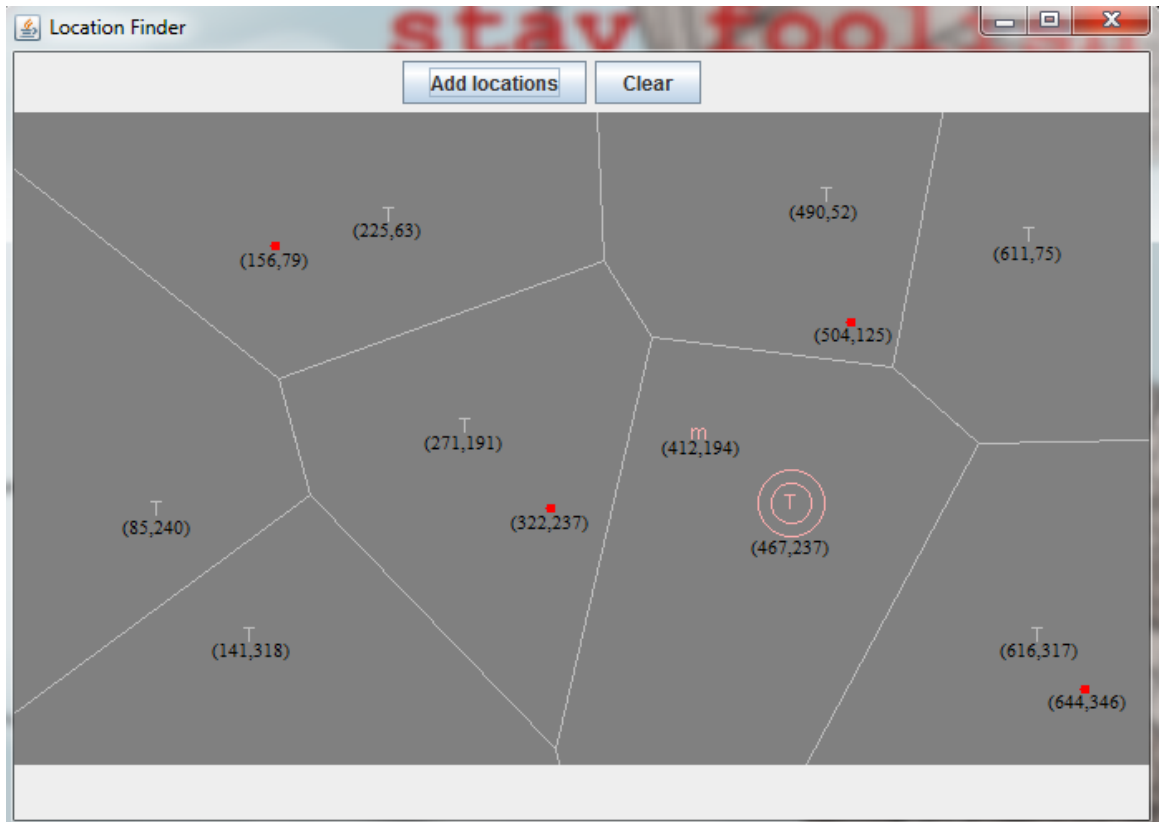


Figure 5.6 Positions of facilities and customers for test case 2

5.2.1 Verification for test case 2

Table 5.3 Aggregate distance for test case 2

Facility	Distance of customer 1 (156,79)	Distance of customer 2 (322,237)	Distance of customer 3 (504,125)	Distance of customer 4 (644,346)	Aggregate Distance (AD)
T1 (85,240)	175.960	237.019	434.495	568.961	1416.435
T2 (141,318)	239.470	198.298	411.118	503.779	1352.665
T3 (225,63)	70.831	199.211	285.806	505.618	1061.466
T4 (271,191)	160.527	68.680	242.167	403.923	875.297
T5 (467,237)	348.833	145.000	117.953	207.870	819.656
T6 (490,52)	335.090	249.898	74.330	331.892	991.210
T7 (611,75)	455.018	331.308	118.106	273.002	1177.434
T8 (616,317)	517.923	304.690	222.279	40.311	1085.203

Table 5.4 Maximum difference for test case 2

Facility	Distance of Cust 1 (156,79)	Distance of Cust 2 (322,237)	Distance of Cust 3 (504,125)	Distance of Cust 4 (644,346)	Max distance	Min distance	Max Diff (MD)
T1 (85,240)	175.960	237.019	434.495	568.961	568.961	175.960	393.001
T2 (141,318)	239.470	198.298	411.118	503.779	503.779	198.298	305.481
T3 (225,63)	70.831	199.211	285.806	505.618	505.618	70.831	434.787
T4 (271,191)	160.527	68.680	242.167	403.923	403.923	68.680	335.243
T5 (467,237)	348.833	145.000	117.953	207.870	348.833	117.953	230.88
T6 (490,52)	335.090	249.898	74.330	331.892	335.090	74.330	260.76
T7 (611,75)	455.018	331.308	118.106	273.002	455.018	118.106	336.912
T8 (616,317)	517.923	304.690	222.279	40.311	517.923	40.311	477.612

5.2.2 Result of test case 2

The facility T5(467,237) selected by the proposed algorithm works fine here also. It gives the minimum aggregate distance from all the available options. Also it gives the minimum of the maximum difference i.e. it minimizes the maximum possible difference between any two customers. Hence the proposed algorithm works as expected.

5.3 Test case 3

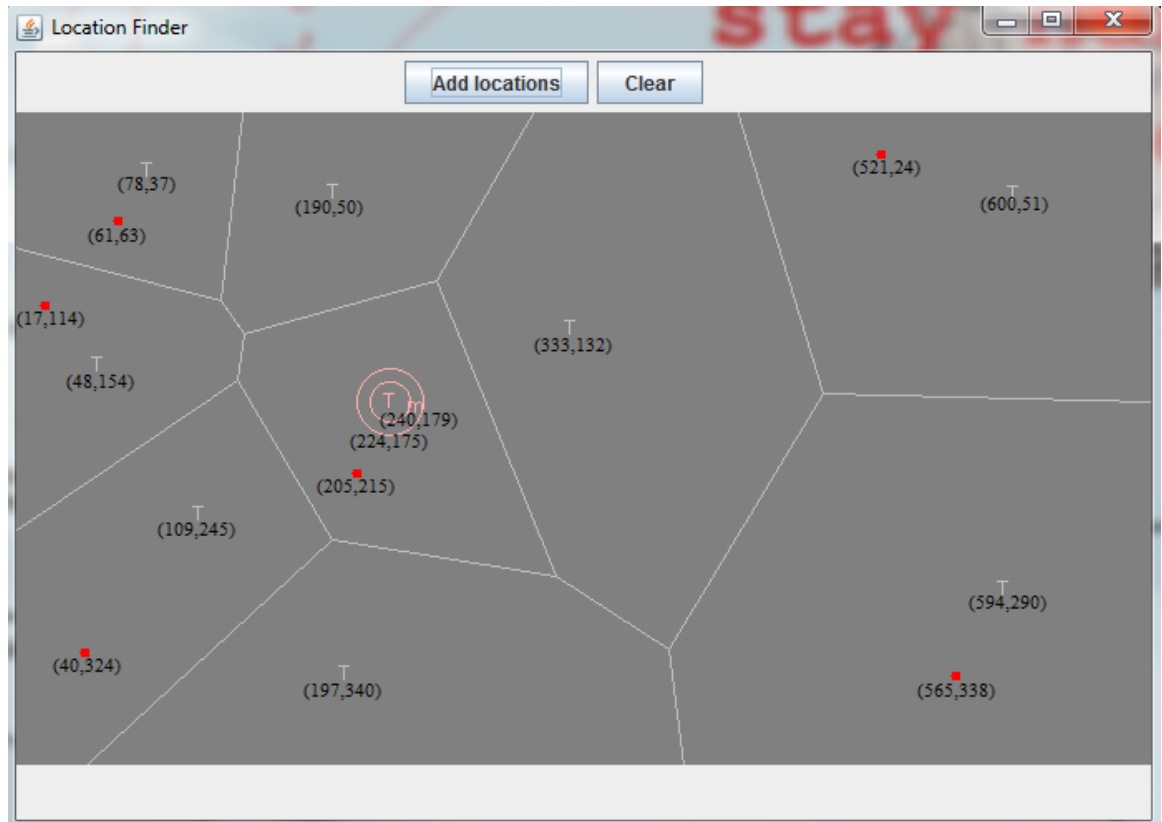


Figure 5.7 positions of facilities and customers for test case 3

5.3.1 Verification of test case 3

Table 5.5 aggregate distances for test case 3

Facility	Distance of cust 1 (17,114)	Distance of cust 2 (40,324)	Distance of cust 3 (61,63)	Distance of cust 4 (205,215)	Distance of cust 5 (521,24)	Distance of cust 6 (565,338)	Agg. Dist (AD)
T1 (48,154)	50.606	170.188	91.924	168.434	490.540	548.767	1520.4 59
T2 (78,37)	98.234	289.505	31.064	218.662	443.191	572.512	1653.1 68
T3 (109,245)	160.078	104.890	188.223	100.578	467.531	465.387	1486.6 87
T4 (190,50)	184.459	312.372	129.653	165.680	332.020	472.831	1597.0 15

T5 (197,340)	288.922	157.813	308.586	125.256	452.584	368.005	1701.1 66
T6 (224,175)	215.801	236.764	197.770	44.28344	333.182	377.955	1405.7 55
T7 (333,132)	316.512	350.304	280.615	152.555	216.813	310.258	1627.0 57
T8 (594,290)	603.245	555.042	579.326	396.164	275.835	56.080	2465.6 92
T9 (600,51)	586.394	623.000	539.133	427.693	83.486	289.126	2548.8 32

Table 5.6 Maximum Difference for test case 3

Facility	Dist of cust 1 (17,114)	Dist of cust 2 (40,324)	Dist of cust 3 (61,63)	Dist of cust 4 (205,215)	Dist of cust 5 (521,24)	Dist of cust 6 (565,338)	Max dist	Min dist	Max Diff (MD)
T1 (48,154)	50.606	170.188	91.924	168.434	490.540	548.767	548.767	50.606	498.161
T2 (78,37)	98.234	289.505	31.064	218.662	443.191	572.512	572.512	31.064	541.44 8
T3 (109,245)	160.078	104.890	188.223	100.578	467.531	465.387	467.531	100.578	366.95 3
T4 (190,50)	184.459	312.372	129.653	165.680	332.020	472.831	472.831	129.653	343.17 8
T5 (197,340)	288.922	157.813	308.586	125.256	452.584	368.005	452.584	125.256	327.32 8
T6 (224,175)	215.801	236.764	197.770	44.283	333.182	377.955	377.955	44.283	333.67 2
T7 (333,132)	316.512	350.304	280.615	152.555	216.813	310.258	350.304	152.555	197.74 9

T8 (594,290)	603.245	555.042	579.326	396.164	275.835	56.080	603.245	56.080	547.16 5
T9 (600,51)	586.394	623.000	539.133	427.693	83.486	289.126	623.000	83.486	539.51 4

5.3.2 Result of test case 3

Here also the selected facility T6(224,175) is the right choice as it minimizes the aggregate distance and also it minimizes the maximum difference up to third from the minimum which is satisfactory as it is in addition to the minimum of aggregate distances. Hence proposed algorithm works fine in all three test cases including this one.

In today's fast pacing world new technologies and internet are proving very helpful to solve daily life problems. New internet and mobile applications have made life comfortable. Finding a facility near to many customers is such an advantage of new technology and internet era. The essence of all the facility location problems is to determine the location of the facility and the allocation of the demands of customers, under the condition of the minimum of the cost. This work presents an invaluable tool for decision makers to reach facility approximately nearer to all. The ultimate goal is to develop a reliable, effective, and robust system that can be used to support customers arriving at high quality decisions on the most suitable facility locations. The effectiveness of the proposed method has been confirmed from computational experiments. In real location problems, due to subjective judgment, imprecise human knowledge and perception in capturing statistic data, many parameters are of both fuzzily imprecise and probabilistically uncertain information.

In this thesis work an algorithm to tackle nearest facility location for multiple customers has been proposed. It tackles the problem by considering not only the aggregate distances of all customers but also the maximum difference between the farthest customer and nearest customer. It takes time of order $O(n \log n)$ as it is based on voronoi diagram of order $O(n \log n)$ and its own time is of linear order.

Previously work has been done on nearest facility location using different approaches like genetic algorithms, GPU systems, fuzzy logics or approximation. In this work voronoi diagram approach has been used. Further combination of two or more techniques may prove to be advantageous and can bring computation time down. Also more accurate and precise decision can result from the combination of two or more techniques. This kind of application may help in real life as to find any nearest facility and the facility may be anything like hospital, chemist shop, police station or any other emergency destinations.

Future scope

The proposed algorithm can be implemented using hybrid techniques which is the combination of techniques to find the nearest facility. Also voronoi construction can be done using an approach specific to the situation. It can work much faster when tried to implement in distributed environment where local system find out there local optimal and then using these local solution a global solution can be obtained.

- In this presented work, the algorithm has been used in a discrete manner i.e. facilities are placed at discrete location. The given algorithm can be used in a continuous environment where it desired to find the location which is placed in continuous space.
- The proposed algorithm can be used by retail chain owner or any other businessmen to find the location optimal for opening the new branch of their business in a competitive environment.
- Reverse of the given problem can be approached that requires the farthest point or facility to be found out using the same proposed algorithm in reverse manner.
- The proposed algorithm may be used in higher and complex level of problem for e.g. in an environment where there are some type of radiation emitting sources. Now to find the location which will have the maximum effect of radiations or the location this will have the lowest impact of the radiations.
- This thesis work presented the nearest facility location for customer in static environment i.e. both facilities are static and customers are also fixed at particular locations. This method can be extended to work in dynamic environment where either customer are moving or facilities are moving or both are in motion.

References

- [1] de Berg, M. van Kreveld, M. Overmars, M. Schwarzkopf (2000). *Computational Geometry: Algorithms and Applications*. Springer-Verlag.
- [2] F. Aurenhammer. "Voronoi diagrams – a survey of a fundamental geometric data structure". *ACM Computing Surveys*, 23(3):345–405, 1991.
- [3] Geng Zhao; Kefeng Xuan; Rahayu, W.; Taniar, D.; Safar, M.; Gavrilova, M.L.; Srinivasan, B., "Voronoi-Based Continuous k Nearest Neighbor Search in Mobile Navigation," *Industrial Electronics, IEEE Transactions on* , vol.58, no.6, pp.2247, 2257, June 2011.
- [4] Minkyu Lee; Dongsoo Han, "Voronoi Tessellation Based Interpolation Method for Wi-Fi Radio Map Construction," *Communications Letters, IEEE* , vol.16, no.3, pp.404,407, March 2012.
- [5] Vachhani, L.; Mahindrakar, A.D.; Sridharan, K., "Mobile Robot Navigation Through a Hardware-Efficient Implementation for Control-Law-Based Construction of Generalized Voronoi Diagram," *Mechatronics, IEEE/ASME Transactions on* , vol.16, no.6, pp.1083,1095, Dec. 2011.
- [6] F. P. Preparata and M. I. Shamos, *Computational Geometry - An Introduction*, New York: Springer-Verlag, 1985.
- [7] P. J. Green and R. Sibson, "Computing Dirichlet tessellations in the plane", *Comput. J.* vol. 21, pp. 168-173, 1977.
- [8] L. J. Guibas, D. E. Knuth and M. Sharir, "Randomized Incremental Construction of Delaunay and Voronoi Diagrams", *Algorithmica*, vol. 7, pp. 381-413, 1992.
- [9] K. Sugihara and M. Iri, "Construction of the Voronoi Diagram for One Million Generators in Single-Precision Arithmetic", *Proc. IEEE*, vol. 80(9), pp. 1471-1484, 1992
- [10] G. Gowda, D. G. Kirkpatrick, D. T. Lee and A. Naamad, "Dynamic Voronoi Diagrams", *IEEE Trans. Inf. Theory*, vol. IT-29, pp. 724-731, 1983.
- [11] Aggarwar, B. Chazelle, L. J. Guibas, C. O'Dunlaing, and C. K. Yap, "Parallel computational geometry", *Algorithmica*, vol. 3, pp. 293-327, 1988.
- [12] Chow, "Parallel algorithms for geometric problems." Ph.D. dissertation. Dept. Comput. Sci., Univ. of Illinois, Urbana, ILL, 1980.

- [13] F. P. Preparata and M. I. Shamos, *Computational Geometry - An Introduction*, New York: Springer-Verlag, 1985.
- [14] S. Fortune, "A sweepline algorithm for Voronoi diagrams", *Proc. 2nd Annual ACM Symp. on Computational Geometry*, pp. 313-322, 1986.
- [15] Shiode, S.; Kuang-Yih Yeh; Hao-Ching Hsia, "On optimal location for three competitive facilities," *Computers and Industrial Engineering (CIE)*, 2010 40th International Conference on, vol., no., pp.1,5, 25-28 July 2010.
- [16] Friggstad, Z.; Salavatipour, M.R., "Minimizing Movement in Mobile Facility Location Problems," *Foundations of Computer Science*, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on , vol., no., pp.357,366, 25-28 Oct. 2008.
- [17] Jungthirapanich, C.; Pratheepthaweephon, T., "A geographic information system-based decision support system (GISDSS) for facility location," *Engineering and Technology Management*, 1998. *Pioneering New Technologies: Management Issues and Challenges in the Third Millennium. IEMC '98 Proceedings. International Conference on*, vol., no., pp.82, 87, 11-13 Oct 1998.
- [18] Ren Peng; Xu Rui-hua; Qin Jin, "Bi-level Simulated Annealing Algorithm for Facility Location Problem," *Information Management, Innovation Management and Industrial Engineering*, 2008. ICIII '08. International Conference on, vol.3, no., pp.17, 22, 19-21 Dec. 2008.
- [19] Zheng Hong-Zhen; Huang Jun-Heng; Zhan De-Chen, "Facility Location Optimization Methods Based on Aggregate and Disperse Moves," *Fuzzy Systems and Knowledge Discovery*, 2007. FSKD 2007. Fourth International Conference on, vol.4, no., pp.669, 673, 24-27 Aug. 2007.
- [20] Takano, Y.; Nishi, T.; Inuiguchi, M., "Facility location and distribution planning with multiple transport alternatives," *SICE Annual Conference (SICE)*, 2011 Proceedings of, vol., no., pp.967, 972, 13-18 Sept. 2011.
- [21] Li Xia; Yanjia Zhao; Ming Xie; Jinyan Shao; Jin Dong, "Mixed integer programming based nested partition algorithm for facility location optimization problems," *Service Operations and Logistics, and Informatics*, 2008. IEEE/SOLI 2008. IEEE International Conference on, vol.2, no., pp.2375, 2381, 12-15 Oct. 2008.
- [22] Shuming Wang; Watada, J.; Pedrycz, W., "Fuzzy random facility location problems with recourse," *Systems, Man and Cybernetics*, 2009. SMC 2009. IEEE International Conference on, vol., no., pp.1846, 1851, 11-14 Oct. 2009.

- [23] Tong Li; Yingtao Li; Zhongtuo Wang, "Application of Plant Growth Simulation Algorithm on Solving Discrete Facility Location Weber Problem," *Innovative Computing Information and Control*, 2008. *ICICIC '08. 3rd International Conference on*, vol., no., pp.145, 145, 18-20 June 2008.
- [24] Yu Dejian; Zhou Dequn; He Xiaorong, "A weighted grey target theory-based strategy model for emergency facility location," *Grey Systems and Intelligent Services*, 2009. *GSIS 2009. IEEE International Conference on*, vol., no., pp.1158, 1162, 10-12 Nov. 2009.
- [25] Shishebori, D.; Mahnam, M.; Nookabadi, A.S., "An efficient approach to discrete Multiple Different Facility Location Problem," *Service Operations and Logistics, and Informatics*, 2008. *IEEE/SOLI 2008. IEEE International Conference on*, vol.2, no., pp.2519, 2524, 12-15 Oct. 2008.
- [26] Bin Yi; Rongheng Li, "Approximation algorithm for the k- product uncapacitated facility location problem," *Computer Science and Information Technology (ICCSIT)*, 2010 *3rd IEEE International Conference on*, vol.5, no., pp.602, 605, 9-11 July 2010.
- [27] Velazquez, E.; Santoro, N., "Distributed Facility Location for Sensor Network Maintenance," *Mobile Ad-hoc and Sensor Networks*, 2009. *MSN '09. 5th International Conference on*, vol., no., pp.237, 243, 14-16 Dec. 2009.
- [28] Diabat, A.; Abdallah, T.; Al-Refaie, A.; Svetinovic, D.; Govindan, K., "Strategic Closed-Loop Facility Location Problem With Carbon Market Trading," *Engineering Management, IEEE Transactions on*, vol.60, no.2, pp.398,408, May 2013.
- [29] Ling Hu; Wei-Shinn Ku; Bakiras, S.; Shahabi, C., "Spatial Query Integrity with Voronoi Neighbors," *Knowledge and Data Engineering, IEEE Transactions on*, vol.25, no.4, pp.863,876, April 2013.