COMPUTING BEST COVERAGE PATH IN THE PRESENCE OF OBSTACLES IN WIRELESS SENSOR NETWORKS

by

Senjuti Basu Roy

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2007

ACKNOWLEDGMENTS

I would like to thank my advisors Dr. Sajal K. Das and Dr. Gautam Das for giving me the guidance that have been instrumental in shaping my career. Without their invaluable advice and assistance, it would not be possible for me to complete this thesis. I would also like to express my gratitude to Dr. Bob Weems for serving on my committee.

I would like to acknowledge the Computer Science and Engineering Department at the University of Texas at Arlington for providing me with much needed financial support. I sincerely acknowledge my research lab, Center for Research in Wireless Mobility and Networking (CReWMaN) for providing me with an intellectual environment to pursue high quality research. I am also greatly indebted to many teachers starting from my high school. Sincere thanks are due to all my friends for their constant encouragements and support. Most importantly, I take this opportunity to express gratitude to my late mother for her relentless support and inspiration in my life. I also sincerely acknowledge my father and relatives in India, for staying beside me at all times and helping me realize my ambitions.

April 2, 2007

ABSTRACT

COMPUTING BEST COVERAGE PATH IN THE PRESENCE OF OBSTACLES IN WIRELESS SENSOR NETWORKS

Publication No. _____

Senjuti Basu Roy, M.S.

The University of Texas at Arlington, 2007

Supervising Professors: Sajal K. Das, Gautam Das

Given a set $S = \{S_1, \ldots, S_n\}$ of n homogeneous wireless sensors deployed in a two dimensional area, a source point s and a destination point t, the *least protected point* p along a path $P(s,t)$ is that point such that the Euclidean distance between p and its closest sensor node S_i is maximum. This distance between p and S_i is called the *Cover* value of the path $P(s,t)$. The *Best Coverage Path* between s and t, denoted as $BCP(s,t)$, is the path that has the minimum cover value. Although there exists efficient algorithms to compute BCP in $O(n \log n)$ time, the presence of *obstacles* inside the two dimensional area has not been addressed in the literature. In this thesis, we consider the problem of computing $BCP(s,t)$ in the presence of m line segment obstacles distributed among n sensors. Because of the presence of obstacles, sensing by sensors can get obstructed and the constructed path may have to detour which pose significant challenges. We propose three algorithms to compute two different variations of the $BCP(s,t)$ problem in the presence of obstacles. In particular, for the case of em opaque obstacles, i.e., which obstruct both the sensing as well the computed path, we develop an algorithm that computes $BCP(s,t)$ in $O((m^2n^2 + n^4)\log(mn + n^2)))$ time. The underlying idea is to

leverage the concept of a quartic-time *Constrained and Weighted Voronoi diagram* among obstacles and creating its dual. For the case of *transparent obstacles* that cannot obstruct sensing but the computed path has to avoid obstacles, we develop two algorithms that compute $BCP(s,t)$. The first algorithm runs in $O(nm^2+n^3)$ time using the *visibility graph* data structure, while the second one is an *approximation* algorithm and requires $O(nm+n^2)$ time using *spanners* of the visibility graph. The approximation factor of the cover value is $O(nk)$ where k is the *stretch factor* of the spanner graph. The proofs of correctness of the three proposed algorithms are also presented in this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Overview

This thesis primarily investigates the modeling of *obstacles*, present in the sensing field and its impact in the computation of coverage path in Wireless Sensor Networks(WSNs). The solutions of these problems are proposed using techniques from *Computational Geometry*, *Graph Theory* and *Approximation algorithm*.

A Wireless Sensor Network(WSN) consists of a set of small untethered sensor nodes that are deployed either randomly or according to some pre-defined statistical distribution, over a geographical region of interest to monitor a physical phenomenon. Each of the sensor node can sense and actuate, compute and communicate wirelessly with a certain confidence level. Physical phenomenon to be sensed can be light, sound, vibration, temperature, pressure, and so on. A sensor has severe resource constraints such as limited signal processing, computation and communication capabilities, storage and energy(battery power). Thus lifetime of a sensor node is typically driven by its battery life.

The applications of WSNs comprise of but not limited to security surveillance in military and battle-fields, emergency alert, chemical and biological sensing, earthquake emergencies, vehicle tracking, traffic control, smart homes and offices, improved health care, industrial diagnosis, and so on. For example, in emergency alert scenario, sensors identify early signs of fire in forests, help fire fighters predict the direction in which fire expands, prevent fire fighters from getting trapped, etc. Not only that, the recent trend to integrate wireless networking into interactive devices such as PDAs, cellular phones, and portable computers has led to foster a wide class of interactive ubiquitous computing applications using sensors.

In order to accomplish the designated task successfully, sensors need to actuate, compute and disseminate the acquired information amongst themselves. Intuitively, *coverage* denotes the quality of sensing of a sensor node. While a sensor senses, it needs to communicate with its neighboring sensor nodes in order to disseminate the acquired data. That is where *connectivity* comes into place. In fact, coverage and connectivity together measures the quality of service(QoS) of a sensor network.

Coverage and connectivity in wireless sensor networks are not unrelated problems. Therefore, the goal of an optimal sensor deployment strategy is to have a globally connected network, while optimizing coverage at the same time. By optimizing coverage, the deployment strategy would guarantee that optimum area in the sensing field is covered by sensors, as required by the underlying application. Whereas by ensuring that the network is connected, it is ensured that the sensed information is transmitted to other nodes and possibly to a centralized base station (called sink) which makes valuable decisions for the application.

Many recent and ongoing research in sensor networks focus on optimizing coverage and connectivity by optimizing node placement strategy, minimizing number of nodes to guarantee required degree of coverage, maximizing network lifetime by minimizing energy usage, computing the most and least sensed path in the given region, and so on. To solve these optimization problems related to coverage, existing research uses mostly probabilistic techniques based on random graph theory, randomized algorithm, computational geometry, and so on. We discuss the current state of the art at this context into more detail in the related work chapter (Chapter 3).

Of particular interest to us is the problem of computing the *Best Coverage Path* (BCP), where given a set of homogeneous sensors deployed in a field and the initial and final locations of an agent that needs to move through the field, determine the path that is most protected by the sensors.

More formally, let $S = \{S_1, \ldots S_n\}$ be a set of n homogeneous wireless point sensors deployed in a 2D sensor field Ω. Each sensor node has the capability to sense data (such

as temperature, light, pressure and so on) in its vicinity (defined by its *sensing radius*). For the purpose of this thesis, assume that these sensors are *guards* that can protect any object within their sensing radius, except that the level of protection decreases as the distance between the sensor and the object increases. Let $P(s,t)$ be any path between a given a source point s and a destination point t. The *least protected point* p along $P(s,t)$ is that point such that the Euclidean distance between p and its closest sensor S_i is maximum. This distance between p and S_i is the known as the *Cover* value of the path $P(s,t)$. The *Best Coverage Path* between s and t, $BCP(s,t)$, is the path that has the minimum cover value. Likewise, the *Worst Coverage Path* (WCP) is one which is least protected by the sensors (e.g., a path that an intruder is likely to follow). These paths are also known as maximal support path (MSP) and maximal breach path (MBP) respectively.

During our investigation, we have seen that *cover* is a new metric and essentially holds all metric properties. To prove this, *cover* holds: (i) *Symmetric Property*, i.e., $cover(x,y) = cover(y,x) \forall x \neq y.$ and (b) *the Triangle Inequality Property*, i.e., $cover(x,y) \leq cover(x,z) + cover(z,y)$. Property (b) is easy to prove since the best observed path to go from x to y has cover $cover(x,y)$ which is $\leq$ the cover value of any other alternative path from x to y. Hence the proof.

As an example, in Figure 1.1, the solid line path depicts the BCP between a source and destination point in the sensor field. In contrast to that, the dotted line path in the Figure 1.1 goes from $source - P - Destination$, but this alternative path is not a BCP.

Existing solution for BCP [13, 46, 38] uses computational geometry based data structures. To solve BCP, the papers [13, 46, 38] use Voronoi diagram, Delaunay Triangulations and even the sparse subgraphs of Delaunay Triangulations. A normal Voronoi diagram is the planar partition of a set $V = \{v_1, v_2, \ldots, v_n\}$ of point site into n convex polygons, with the property that all points lying inside a particular convex polygon is closer to its own point site v_i of V than any other point sites. This BCP problem is solved by creating Delaunay Triangulations, which is the dual of Voronoi diagram. It is

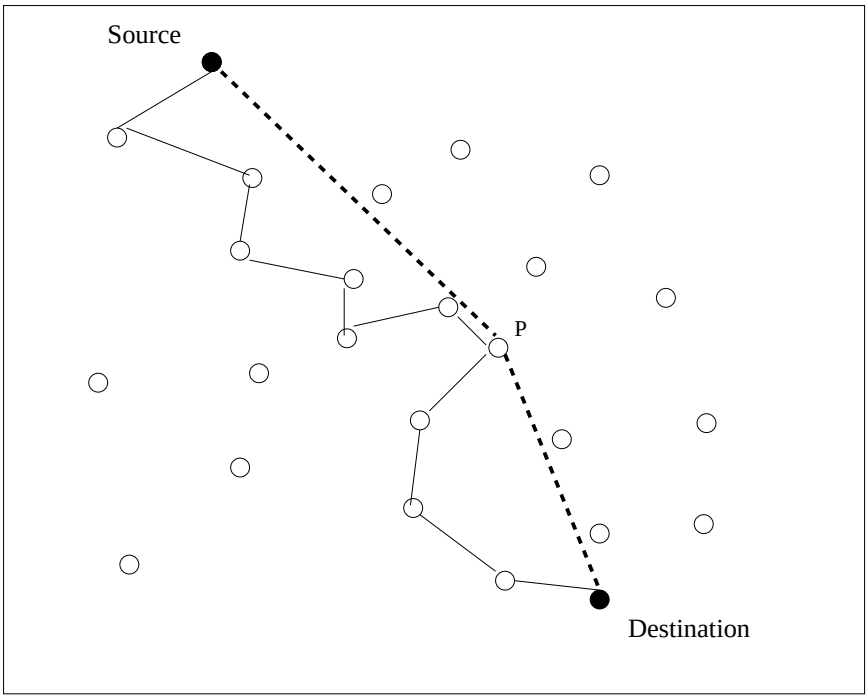

Figure 1.1. The BCP amidst a set of sensors.

also proved that at least one of the BCPs follow only the edges of Delaunay Triangulation. Furthermore, [46] shows that sparse subgraphs of the Delaunay triangulation, such as Gabriel Graphs and even Relative Neighborhood Graphs also contain BCP. These graphs can be efficiently computed in $O(n \log n)$ time since the Voronoi diagram is a linear-sized structure. We will discuss these data structures individually and in more detail in the Chapter 2.

However, in solving these problems, existing literature has not considered the presence of *obstacles* in the sensor field, i.e., objects that obstruct paths and/or block the line of sight of sensors. The presence of obstacles is especially realistic in a random dense deployment of sensors in unmanned terrain, e.g., buildings and trees, uneven surfaces and elevations in hilly terrains, and so on. Hence in this thesis, we initiate a study of the presence of obstacles in computing best coverage paths.

Next, we elucidate why computing BCP in presence of obstacles is a novel and challenging problem. In addition, we discuss several potential application domains where computation of BCP in presence of obstacles is useful.

1.2 Motivation

Our motivation for taking BCP problem is instigated by the possibility of its potential applicability and relevance in several diverse application domains. Some potential applications comprise of but not limited to security and surveillance and QoS of WSN. Not only that, BCP can be a widely studied problem in any motion planning domain, such as Robotics and so on. In this section, we discuss some of these potential applications and challenges that motivated us towards computing BCP problem in presence of obstacles.

1.2.1 Security and Surveillance Aspect

As stated earlier, coverage and connectivity together ensures the QoS of the network. Computing BCP has a great impact in security surveillance and can be appropriate in military and emergency control related applications. For example, if the given domain is under surveillance by the sensors which are capable of detecting, say fire, then clearly BCP is the most preferred path of traversal of an agent who is moving around the field gathering critical information. A related problem in this fire detection sensor network example is, one may ask how well the network can observe a given area and what the chances are that a fire starting in a specific location will be detected in a given time frame. Furthermore, coverage formulations can try to find weak points in a sensor field and suggest future deployment or reconfiguration schemes for improving the overall quality of service. On the contrary, WCP (worst coverage path) would be the preferred path of adversaries in the battle-field.

1.2.2 Challenges Posed by Obstacles

Prior work on computing BCP relied on the use of linear-sized Voronoi Diagram, Delaunay Triangulations, and so on. However, it is not easy to extend these ideas for solving BCP with obstacles. During our investigation, we have seen that to solve the BCP problem with obstacles, the existing ideas has to be considerably and non-trivially

extended. Moreover, we also showed that the *visibility graph* - a popular structure extensively used in shortest path computations in the presence of obstacles [33] - is also not applicable for solving BCP problem for opaque obstacles, since best coverage paths need not follow edges of the visibility graph. In fact, to solve the BCP problem for opaque obstacles, we have developed an algorithm that takes quartic-time, based on constructing a specialized dual of the *Constrained and Weighted Voronoi Diagram* [9] among obstacles. In our case, this structure is only *constrained* because of the presence of obstacles in the domain (but all point sites are of equal weights). So, henceforth we will call this *Constrained and Weighted Voronoi Diagram* as *Constrained Voronoi Diagram*, a more appropriate terminology to our problem scenario. While the time complexity using this diagram is obviously high, we conjecture that this is likely to be optimal. However, to solve BCP problem for transparent obstacles, we have shown that the best coverage path is contained in the visibility graph (with suitably defined edge weights), and thus we could develop more efficient exact and approximation algorithms for it. In fact, our approximation algorithm is based on the notion of *k-spanners* [18] which are approximation structures of the visibility graph.

1.2.3 Path Planning Problems

As the calculation of the *best coverage path* is essentially a path planning problem, we believe BCP has general applications even in path planning of wireless sensor nodes. Our algorithm gives the *best* obstacle- free path where the constructed path stays as close as possible to the protectors (sensor nodes). With the latest advent of sensor technology, where mobile sensors are easily available, we can even think of an complementary application scenario, where a set of mobile sensor nodes (for example fuel tanks) move along the BCP, to stay as close as possible to the protectors, such as, fire stations.

Other potential application of BCP might occur in the field of robotics where motion planing problems are widely studied.

Next we formally define the problems.

1.3 Problem Definition

Let $S = \{S_1, \ldots S_n\}$ be a set of n homogeneous wireless point sensors deployed in a 2D sensor field Ω. Assume that in addition to the n sensors, there are also m *line segment obstacles* $O = \{O_1, \ldots, \emptyset_m\}$ placed in the sensor field. Line segments are fundamental building blocks for obstacles, as more complex obstacles (e.g., polygonal obstacles) can be modeled via compositions of line segments. We consider two types of obstacles: (a) *opaque obstacles* which obstruct paths as well as block the line of sight of sensors, and (b) *transparent obstacles* which obstruct paths, but allow sensors to "see" through them. Examples of the former may include buildings - they force agents to take detours around them as well as prevent certain types of sensors (such as cameras) from seeing through them - while examples of the latter may include lakes - agents have to take detours around them but cameras can see across to the other side. When obstacles are opaque, we refer to the best coverage path problem as *BCP for Opaque Obstacles*, whereas if the obstacles are of the transparent type we refer to the problem as *BCP for Transparent Obstacles*. Figure 1.2 is an example of a $BCP(s,t)$ amidst two sensors and four opaque obstacles, whereas Fig 1.3 shows the $BCP(s,t)$ in the same sensor field but assumes the obstacles are transparent.

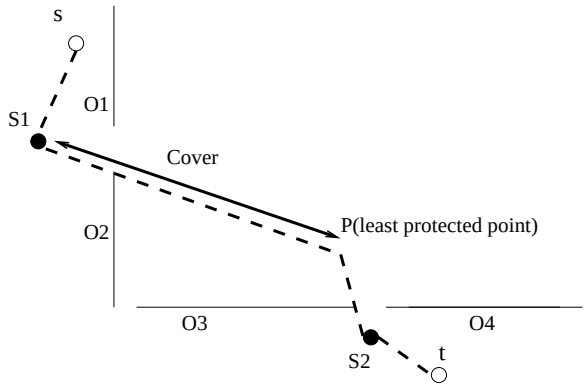

Figure 1.2. $BCP(s,t)$ for Opaque Obstacles.

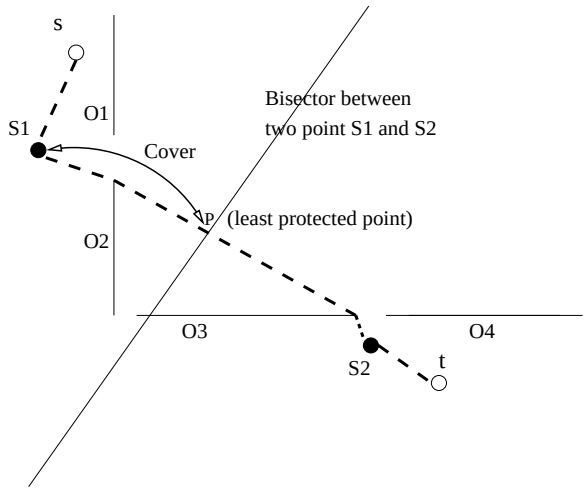

Figure 1.3. $BCP(s,t)$ for Transparent Obstacles.

We assume that each static sensor node knows its position information, either through a low-power Global Position System (GPS) receiver or through some other approach.

Notations and Terminologies

The associated notations and terminologies are stated below:

$||xy||$ denotes the Euclidian distance between two points x and y.

The distance of a point x to a point set V is the smallest distance of x from all the points in V. Defining mathematically, $dist(x,V) = min_{y \in V}||xy||$. The point set V may be infinite. In fact, in our application V is infinite because V is the set of points in the path in the domain which is the best coverage path.

The coverage distance of a point set U by a point set V can be defined as *Cover(U, V)*, the maximum of the minimum distance of every point of U to all the points of V. Mathematically,
$Cover(U,V) = max_{x \in U} dist(x,V)$. or $Cover(U,V) = max_{x \in U}(min_{y \in V}||xy||)$. Clearly this is a max-min problem.

A path from s to t, denoted as $P(s,t)$, that achieves the minimum coverage distance, *Cover*$(P(s,t),S)$ is known as the *Best Coverage Path*, $BCP(s,t)$.

Next we state the problem definitions:

Problem Statement

Inside a two-dimensional domain Ω, with a set $S = \{S_1, \ldots, S_n\}$ of n sensor nodes and $O = \{o_1, \ldots, o_m\}$ of m obstacles, given a starting point s and ending point t inside the domain, compute the *Best Coverage Path*, $BCP(s,t)$, so that the cover value of the path $P(s,t)$, i.e., *Cover(P(s,t), S)* is minimized. We compute $BCP(s,t)$ in two following cases:

- obstacles are opaque - The corresponding problem in known as $BCP(s,t)$ for Opaque Obstacles.
- obstacles are transparent - The corresponding problem in known as $BCP(s,t)$ for Transparent Obstacles.

1.4 Contributions of this Thesis

Our contributions in this thesis may be summarized as follows:

- We have initiated a study of the presence of *obstacles* and their impact in the computation of Best Coverage Paths. We have developed two variants of the problem, the BCP problem for opaque obstacles and the BCP problem for transparent obstacles, based on variants of obstacle properties. We have shown that obstacles significantly complicate these problems.
- We have designed an $O((m^2n^2 + n^4)\log(mn + n^2)))$ time algorithm for BCP problem, given n sensor nodes and m opaque line obstacles. The algorithm is based on constructing a special dual of the Constrained Voronoi diagram, and the time complexity is dominated by the quartic combinatorial structure of the diagram itself.
- We have designed an $O(nm^2 + n^3)$ time algorithm for the BCP problem, given n sensor nodes and m transparent line obstacles. The algorithm is based on constructing the visibility graph of the sensors and obstacles (with special weights assigned to edges).

- We have also designed an $O(nm + n^2)$ time approximation algorithm for the BCP problem for transparent obstacles. This algorithm is based on constructing a k-spanner of the visibility graph. The approximation factor in the cover of the computed path is $O(nk)$, where k is the Euclidean stretch factor of the spanner.

1.5 Organization of the Thesis

The rest of the thesis is organized as follows:

In Chapter 2, we illustrate in detail the *Coverage and Connectivity* in wireless sensor network (section 2.1 and 2.2). Mathematical model of sensing and communication are also discussed in conjunction to that. In Section 2.3, we expand the idea of related computational geometry tools and techniques that we have used in this thesis. Precisely, Chapter 2 details out all the preliminaries that are important in order to understand our proposed solution.

Chapter 3 illustrates the current state of the art research in coverage problems in WSNs. In particular, we classify the related coverage research in three broad category and expand each of these categories in more detail in terms of the problem definition, solution technique and performance issues.

In chapter 4, we discuss BCP problem for opaque obstacles in detail. We outline our proposed algorithm intuitively, state the algorithm, provide the proof of correctness and analyze its running time. In addition to that, dual creation and edge weighing process (two important parts of our proposed solutions) are also discussed in detail.

Likewise in Chapter 5, we discuss the BCP problem for transparent obstacles. In particular, we individually discuss the exact and approximation algorithm for computing BCP for transparent obstacles. We discuss the algorithm, provide proof of correctness and analyze running time for both the exact and approximation algorithm. In addition, we calculate the approximation factor for approximation algorithm.

Finally in Chapter 6 we conclude with some directions of future research work.

CHAPTER 2

PRELIMINARIES

This chapter elucidates the preliminary concepts related to Coverage and Connectivity in WSN and Computational Geometry. In Sections 2.1 and 2.2, we discuss the concepts of Coverage in WSN along with modeling coverage and communication model mathematically. Section 2.3 expands the related computational geometry concepts.

2.1 Concepts of Coverage and Connectivity

Optimal resource management and assuring reliable QoS are two of the most fundamental requirements in wireless sensor networks. To provide better QoS, sensor deployment strategies play a very important role, because they relate to the issue of how well each point in the sensing field is covered. However, due to severe resource constraints and hostile environmental conditions, it is non-trivial to design an efficient deployment strategy that would minimize the cost, reduce computation, minimize node to node communication and provide a high degree of area coverage, while at the same time maintain a globally connected network. Challenges also arise because of the fact that often topological information about a sensing field is unavailable and such information may change over time. Many wireless sensor network applications are required to perform certain functions that can be measured in terms of *area coverage*. In these applications, it is necessary to define precise measures of efficient coverage that will impact overall system performance. In [15] three types of coverage are broadly defined:

1. *Blanket Coverage*: To achieve a static arrangement of sensor nodes that maximizes the detection rate of targets appearing in the sensing field.
2. *Barrier Coverage*: To achieve a static arrangement of sensor nodes that minimizes the probability of undetected penetration through the barrier.

3. *Sweep Coverage*: To move a number of sensor nodes across a sensing field, such that it addresses a specified balance between maximizing the detection rate and minimizing the number of missed detections per unit area.

It is interesting to notice that the area coverage problem is closely related to a classical problem of computational geometry, namely the *Art Gallery Problem* [25, 27]. This problem seeks the minimum number of guards needed in the museum so that every point in the museum is protected by at least one guard. This is similar to the coverage problem which tries to find out the minimum number of sensors so that every point in the field is protected. An early result in art gallery research, due to V. Chvátal [25, 27], asserts that $\lfloor \frac{n}{3} \rfloor$ guards are occasionally necessary and always sufficient to guard an art gallery represented by a simple polygon of n vertices. Since then, numerous variations of the art gallery problems have been studied, including mobile guards, guards with limited visibility or mobility, guarding of rectilinear polygon and so on. The main difference between the the art gallery problems and the BCP problems in this thesis is that the former problems (such as Watchman Route, Robber Route [25, 27] and so on) attempt to determine paths that minimize total Euclidean distances under certain constraints, whereas the metric to be minimized in the latter problems (the *cover* of the path) is sufficiently different from Euclidean distance, thus requiring different approaches.

2.2 Modeling of Sensing, Coverage and Connectivity

In this Section, we describe the sensing model and definitions of coverage and connectivity. These are important to our work since the modeling of sensing is required to understand the concept of coverage.

2.2.1 Sensing Model

The *sensing intensity*, S, of a sensor s_i at point P is defined as [39]:

$$S(s_i, P) = \frac{\lambda}{[d(s_i, P)]^{\gamma}} \tag{2.1}$$

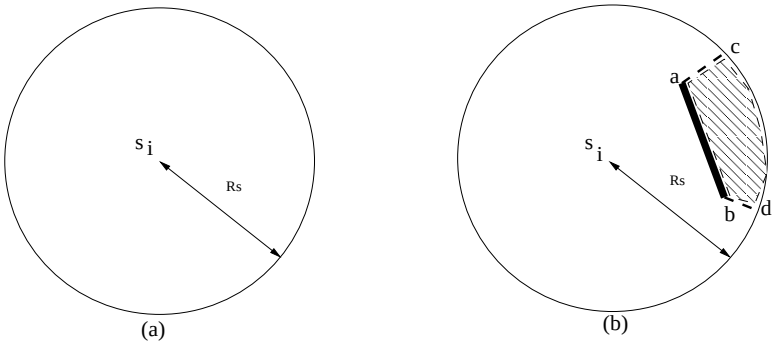

Figure 2.1. The sensing Radius R_s takes an irregular shape because of obstacle.

In this definition, λ and γ are (positive) sensor dependent parameters and $d(s_i, P)$ is the Euclidean distance between the sensor and the point. Typically the value of γ is dependent on environmental parameters and varies between 2 to 5. Since the sensing intensity rapidly decreases as the distance increases, a maximum sensing range for all sensors can be defined. It is customary to assume a *binary* sensing model, according to which a sensor is able to sense from all the points that lie within its sensing range. Any point lying beyond is considered outside its sensing range. Thus, according to this model, the sensing range for each sensor is confined within a circular disk of radius R_s. In a heterogeneous sensor network, the sensing radii may vary for different sensors. In this work, we have assumed *homogeneous* sensing model with a homogeneous sensing radius, R_s. Implementing our algorithm for non-homogeneous sensor nodes is a trivial extension and is omitted from this thesis. In general, the homogeneous or nonhomogeneous sensors ideally produce circular sensing radii. However, due to the presence of the obstacles, the sensing radii gets irregular shape and hence the computation of $BCP(s, t)$ becomes computationally harder to solve.

As an example, consider Figure 2.1(a), which shows a perfect circular sensing radius R_s which is not the case for the Figure 2.1 (b). Due to the presence of the obstacle ab, the region $abdc$ can not be seen by the sensor s, and thus creates a hole in the sensing region. This is how the sensing region becomes irregular in shape.

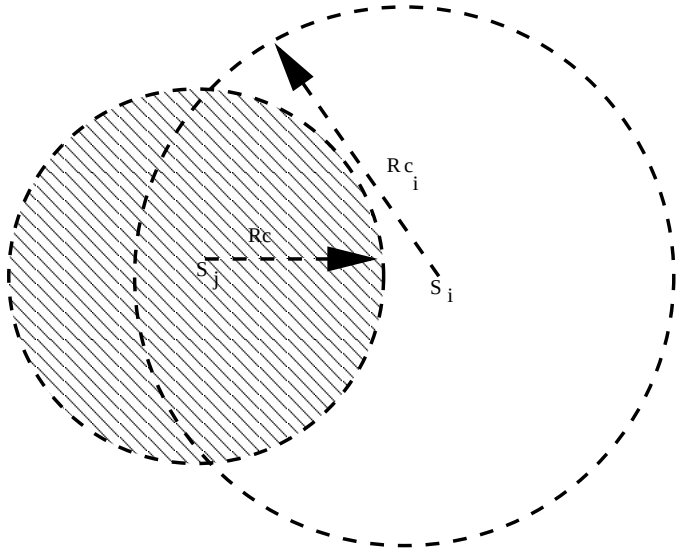

Figure 2.2. Sensors s_i can only communicate with s_j but s_j can not since $d(s_i, s_j) > min(Rc_i, Rc_j)$.

2.2.2 Communication Model

Like sensing, the communication relationship between two sensors can also be modeled mathematically.Two sensors s_i and s_j are able to communicate with each other if the Euclidean distance $d(s_i, s_j)$ between them is less than or equal to the minimum of their communication radii, Rc_i and Rc_j respectively, i.e., when $d(s_i, s_j) \leq min(Rc_i, Rc_j)$. This essentially means that the sensor with smaller communication radius falls within the communication radius of the other sensor. Two such nodes that are able to communicate with each other are called 1-hop neighbors. The communication radii might vary depending on the residual battery power (energy) of an individual sensor. In this thesis, we assume that the communication radii for all the nodes are the same, denoted by R_c. Usually, communication is more than the sensing radius.

In homogeneous sensor networks, all sensors are ideally of equal communication radii; hence they can communicate with each other. In contrast to that, as depicted in Figure 2.2, in heterogeneous sensor network, sensors may have varying sensing and communication radius. For this case, it might be possible that a sensor with communication radius Rc_i is able to communicate with sensor of sensing radius Rc_j, where $Rc_i > Rc_j$

but Rc_j can not as $d(s_i, s_j) > Rc_j$. Even here, due to the presence of obstacles, the communication region may take irregular shape instead of being circular.

2.2.3 Graph Theoretic Perspective of Coverage Problems

As illustrated in previous Section 3.1, sensors can ideally sense each point inside the sensing disk and communicate(broadcast) with other sensors inside the communication radius. By a simple broadcasting, each node u can gather the location information of all nodes within the transmission range of u. This communication relationship basically leads to a graph theoretic topology. Comparing it with a standard graph model, the entire network is a graph $G = (V, E)$ where, the set of sensors constitutes the vertex set V and the set E is comprised of the set of links present between the vertices.

Essentially coverage related problem thus can be realized as a graph theoretic problem. Many of the ongoing research on coverage and connectivity manipulates this communication graph and optimize different parameters of the constructed graph, such as, number of nodes, no of links or the topology of the graph itself, and so on.

2.3 Concepts Related to Computational Geometry

In this Section, we discuss the related computational geometry concepts and the various data structures that used in the thesis.

Computational Geometry is the study of the algorithms that manipulates abstract geometric objects, which are themselves used to model the real world.

In this work, we have used a Constrained Voronoi diagram (henceforth known as C-Voronoi Diagram) among obstacles [9, 11]; In addition to that, visibility graph [33], a popular proximity graph based data structure in Euclidean Geometry, is also used in the BCP problem for transparent obstacles. For the approximation algorithm, we have used the concepts of Euclidean Spanners [18, 29] of Visibility Graphs. Not only that, the normal linear sized Voronoi diagram [33], Delaunay Triangulation [33], Constrained Delaunay Triangulation [31] are also studied at the context of study of related work.

Next, we discuss about each of these data structures and its application in our work in different subsections.

2.3.1 Voronoi Diagram

A normal Voronoi diagram [33] is the planar partition of n point sites into n number of convex polygons such that each polygon contains exactly one point site and all points in a given polygon is closer to its generating point site than any other point sites. A Voronoi diagram is sometimes also known as a *Dirichlet tessellation* [33].

Let S denote a set of n points(called *sites*) in the plane. For two distinct sites p,q $\in S$, the *dominance* of p over q is defined as the *subset* of the plane being at least as close to p as to q. Formally, $dom(p, q) = x \in R^2 | d(x, p) \leq d(x, q)$,
where d denotes the Euclidean distance function. Clearly, $dom(p, q)$ is a closed half plane bounded by the perpendicular bisector of p and q. This bisector separates all points of the plane closer to it and will be termed the *separator* of p and q. The region of a site $p \in S$ is the portion of the plane lying in all of the dominance of p over the remaining sites in S.

There exist efficient algorithms to compute Voronoi diagram of a set of n point sites. Using $O(n)$ storage space and $O(n \log n)$ time, the Voronoi diagram can be computed efficiently by divide and conquer or sweep-line algorithm [33].

It is proved in [13, 46, 38], that BCP and WCP problems without obstacles can be solved using Voronoi Diagram and its dual Delaunay Triangulation. Particularly, it is shown that the WCP path follows the edges of the constructed Voronoi Diagram.

In Figure 2.3, a normal Voronoi diagram of a set of 7 arbitrary point sites is shown. Each of these 7 cells are separated from one another by the dotted lines, which actually are the perpendicular bisectors between point sites. This set of bisectors also constitutes the edge set of the Voronoi diagram.

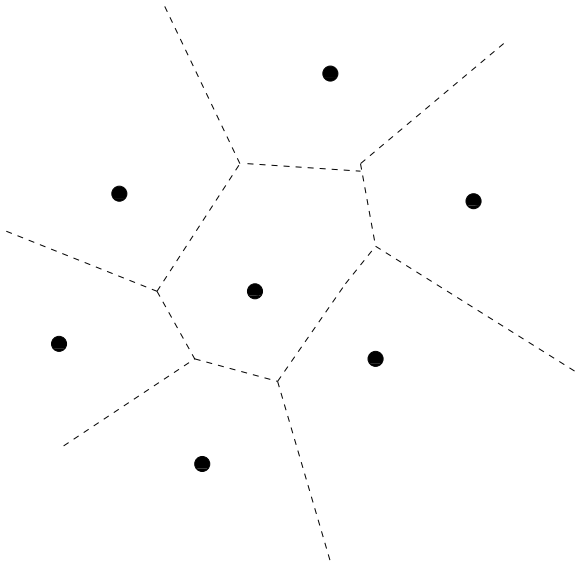

Figure 2.3. A Voronoi diagram of a set of 7 points.

We use this Voronoi diagram example to analyze other proximity graphs; Delaunay Triangulation (DT),Gabriel Graph (GG) and Relative Neighborhood Graph (RNG) to explain sparse subgraph property.

2.3.2 Delaunay Triangulation

Delaunay Triangulation (DT) is a planar graph, dual of the Voronoi Diagram. In DT, there exists an edge between a pair of points if they share a common edge in their Voronoi Diagram and obeys the empty circle property. A triangulation of R is a Delaunay triangulation, denoted by $Del(R)$, if the circumcircle of each of its triangles does not contain any other vertices of S in its interior. It is simple to create a Delaunay Triangulation [33] of any point set P by first creating the Voronoi Diagram of P, then creating the dual graph of this diagram. This computation requires no more than $O(n \log n)$ time. An alternative way to create a DT is to use a randomized incremental algorithm whose complexity is $O(n \log n)$. This algorithm begins with a large initial triangle and incrementally adds each point while maintaining the DT properties.

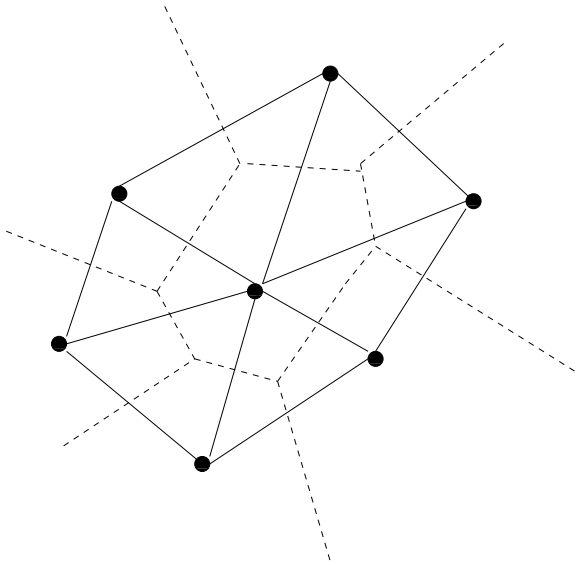

Figure 2.4. The Delaunay Triangulation of the Voronoi diagram of Figure 2.3.

Figure 2.4 contains the DT of the Voronoi diagram of the Figure 2.3. Due to the empty circle property, many possible edges between point sites are removed in DT; hence DT is sparse and has $O(n)$ edges of n point sites.

In [13, 46, 38], it is shown that the BCP without obstacles follows the edges of the DT of the sensor nodes. It is also proved that one of the BCPs is contained only in the edges of DT.

2.3.3 Gabriel Graph

A Gabriel Graph (GG) is a sparse subgraph of Delaunay Triangulation. The Gabriel graph consists of those edges connecting two points of the point set such that the circle whose diameter is the edge does not contain any other points of the point set in its interior. This well known proximity graph establishes neighborhood relationship between objects, for example, points in the Euclidean Space. The formal definition of a Gabriel Graph is as follows: In a point set P, an edge exists between two vertices x and y iff:

$d(x, y) \leq \sqrt{d^2(x, z) + d^2(y, z)}, \forall z \in P, z \neq x, y$

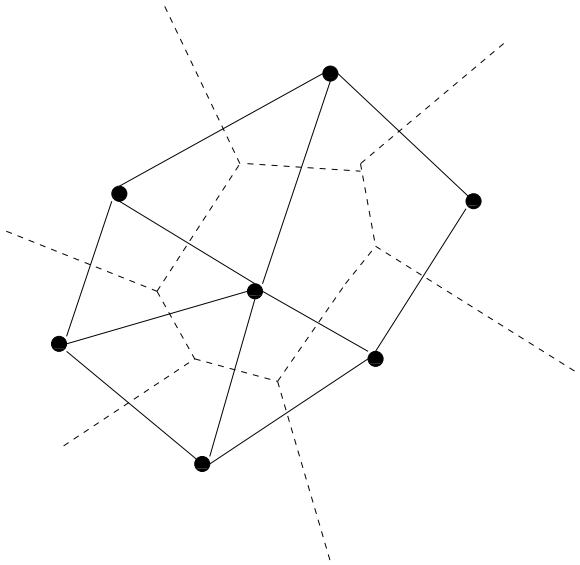

Figure 2.5. The Sparser Gabriel Graph of DT of Figure 2.4.

A Gabriel Graph can be created in $O(n \log n)$ time by first finding the Delaunay Triangulation and Voronoi Diagram for the set of points [33]. Then, for each edge in the triangulation, if the edge intersects its Voronoi edge, it is added as an edge to the GG.

Figure 2.5 contains the GG of the DT of Figure 2.4. It is evident that GG is sparser than DT. In [46], it is shown that the edge set of GG contains the BCP.

2.3.4 Relative Neighborhood Graph

A more sparse subgraph of DT is Relative Neighborhood Graph (RNG) [28] in which an edge exists between two points iff their lune doesn't contain any other nodes inside. Formally, lune is the region of intersection of two points if we draw an arc through each of these points. In 2-dimension, RNG of n points can be computed in $O(n \log n)$ time.

Figure 2.6, is the RNG of the DT of Figure 2.4. Due to the empty lune property, here one more edge is removed from the GG and the RNG graph becomes even more sparse than GG.

The discussions of preceding Subsections 2.3.1, 2.3.2, 2.3.3, and 2.3.4 are related to those data structures which model the BCP and WCP without obstacles. Next we

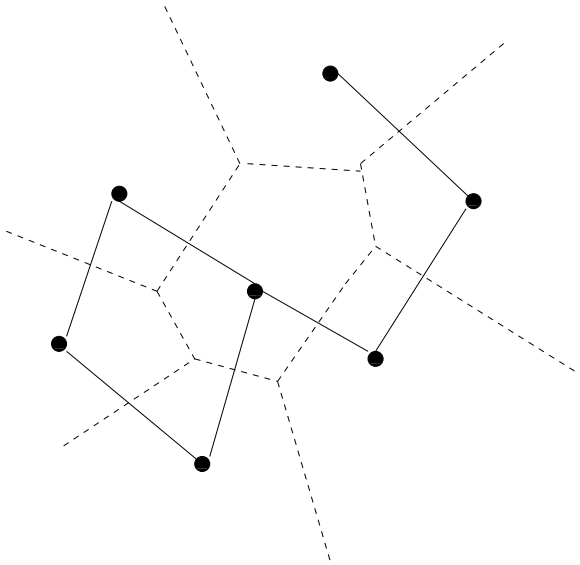

Figure 2.6. Relative Neighborhood Graph of DT of Figure 2.4.

discuss data structures which we have used to model the BCP problem for opaque and transparent obstacles.

2.3.5 Constrained Voronoi Diagram

Apart from sensor network, obstacles are of interest in areas like autonomous vehicle navigation, where hilly terrains are being modeled. Specialized Voronoi diagram are used to model this type of problems. An example of this is Peeper's Voronoi Diagram, where each point in the plane is assigned to the closest visible site, and visibility is constrained to a segment on a line avoiding the convex hull of the sites (Reference Figure2.7) [16]. This structure can attain a size of $\Theta(n^2)$ and is constructible in $O(n^2)$ time. Figure 2.7 depicts the Peeper's Voronoi diagram of four point sites . Edges contributed by visibility rays (or bisectors) are shown by solid (or dashed) lines. Note that here are points not covered by any region. As can be seen from Figure 2.7, unlike a normal Voronoi diagram, the presence of obstacles greatly complicates the diagram, and Voronoi regions of a site may even be disconnected. Not only that, there may be holes not visible from any of the point sites.

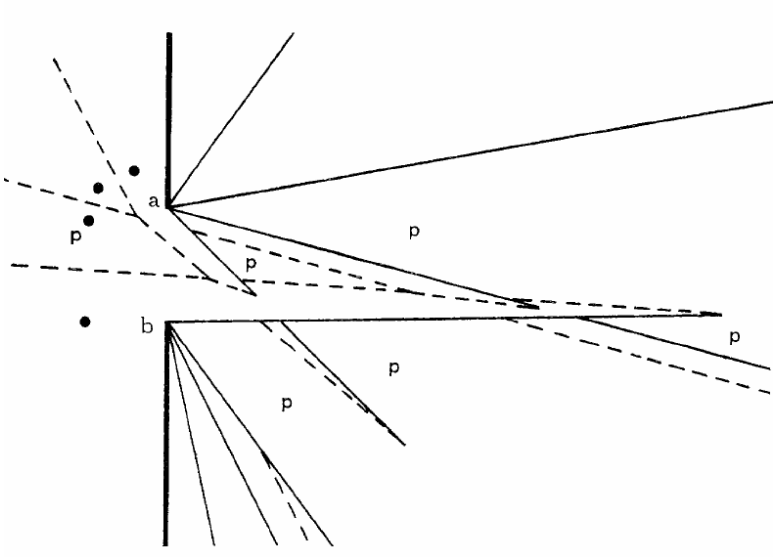

Figure 2.7. Peeper's Voronoi diagram of 4 sites.

As a generalization of Peeper's Voronoi Diagram, *Constrained Voronoi Diagram*, originally named as *Constrained and Weighted Voronoi Diagram* [9] is studied. The existence of obstacles make a simple divide-n-conquer and a sweep algorithm for constructing this diagram incorrect for constructing this diagram. The algorithm for this diagram construction is done [9] by exploiting the property of arrangement of a set of lines, where the lines are completely determined by the sites and obstacles. By constructing a visible-site list and a closest-site list for each edge in the arrangement, this Constrained Voronoi diagram is computed in the original work [9].

There are even planar Voronoi diagrams which model obstacles in the region with certain constraints in the positioning of obstacles. Constrained Voronoi diagram is the classical diagram constrained by a set of line segments [36] in literature. But here, unlike Peeper's Voronoi Diagram, the obstacles are non-crossing line segments and their end points belong to the set of point sites itself. This makes the diagram planar and easy to compute. Plane-sweep techniques can be applied to compute this diagram.

As a generalization to this Peeper's Voronoi diagram, in [9] the authors considered the *Constrained and Weighted Voronoi diagram* over a set of weighted sites amidst a set of line segment obstacles in the plane. Both the Peeper's and the Constrained Voronoi

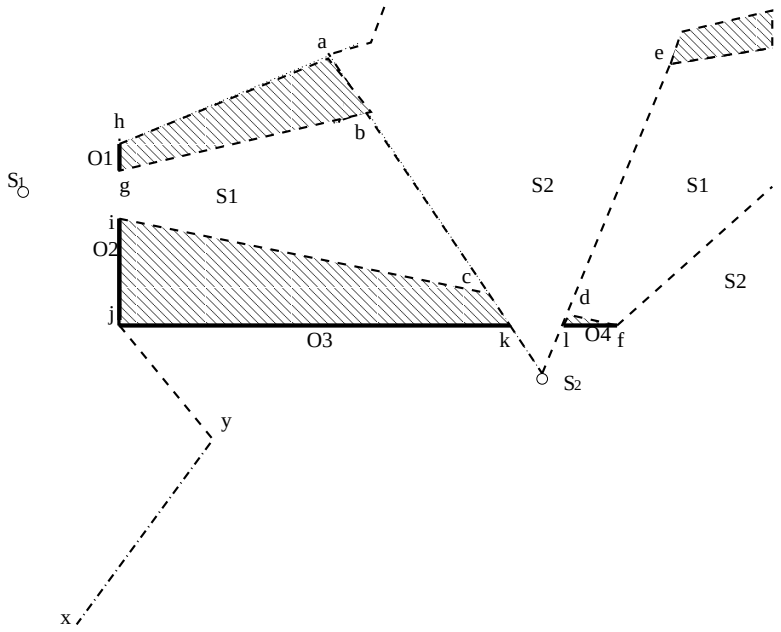

Figure 2.8. C-Voronoi diagram of sensors and obstacles.

diagrams are distinct from other Voronoi diagrams involving obstacles studied in papers such as [30, 4, 36] - the Voronoi diagrams studied in [30, 4, 36] require that every obstacle endpoint be a site, whereas in the Peeper's and Constrained Voronoi diagram the Voronoi sites are distinct from the obstacle endpoints.

As an example, consider Figure 2.8 which shows the C-Voronoi diagram where S_1, S_2 are two sensors and O_1, O_2, O_3, O_4 are 4 obstacles. The filled areas are dark regions which cannot be sensed by either of the two sensors S_1, S_2. The remaining cells of the C-Voronoi diagram are labeled by the sensors to which they are closest. This constrained Voronoi diagram has $\Omega(m^2n^2)$ combinatorial complexity and its construction takes $O(m^2n^2 + n^4)$ time and space. This algorithm is optimal when $m \geq cn$, for any positive constant c.

2.3.6 Visibility Graphs

Given a set of line obstacles, a visibility graph is a graph of intervisible locations [33]. Each node or vertex in the graph represents a location (such as an obstacle endpoint), and each edge represents a visible connection between them (that is, if two

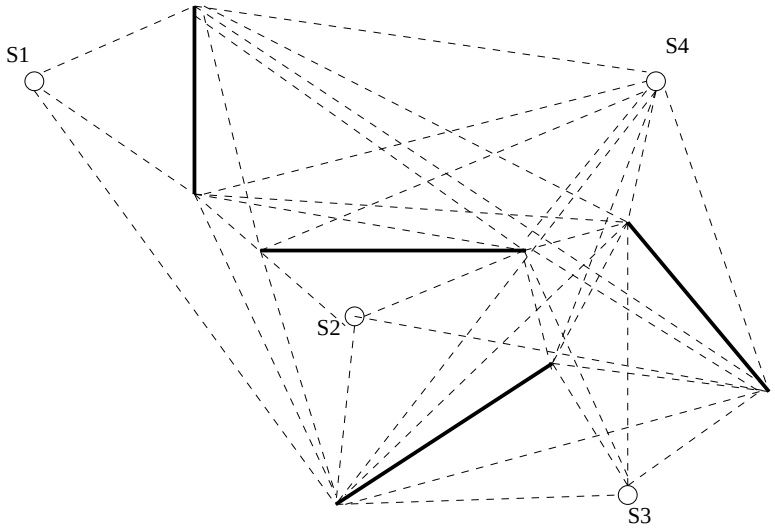

Figure 2.9. Visibility Graph of 4 sensors and 4 line obstacles.

locations can see each other, an edge is drawn between them). This graph gives the shortest obstacle avoiding path between two points in the region. The visibility graph of n sites can have as many as $O(n^2)$ edges. In our problem, we need to consider the visibility graph of n sensors and m obstacles, thus with a total of $n + 2m$ locations. The visibility graph of 4 sensor nodes and 4 line obstacles in shown in Figure 2.9. We use visibility graphs for solving the $BCP(s, t)$ problem for transparent obstacles.

Let $O = o_1, o_2, \ldots, o_n$ denotes the set of disjoint set of obstacles in the plane. The naive way of computing the visibility graph is to find the pair of vertices that can see each other. That means for every pair of vertices, we need to see if the line segment joining them intersects with an obstacle. This test takes $O(n)$ time when done naively, leading to an $O(n^3)$ running time.

However, there exists more efficient algorithms based on arrangements which run in $O(n^2)$ time [43]. Mitchell [26] proposed an algorithmic solution to compute visibility graph breaking the quadratic barrier. Optimal algorithm to compute visibility graph takes $O(n \log n)$ time [24].

A number of even faster algorithms on visibility graph is also known, including the output-sensitive algorithm by [37].

2.3.7 k-Spanner of Visibility Graph

Let, $G = (V, E)$ be a connected n-vertex graph. A subgraph $G' = (V, E')$ is a k-*spanner* if, between any pair of vertices, the distance in G' is at most k times longer than the distance in G. The value of k is the *stretch factor* associated with G'. To solve the $BCP(s, t)$ problem for transparent obstacles, we have investigated the available results, and have used the k-spanner of the visibility graph [29] for the approximation algorithm. This spanner has linear number of edges, and can be constructed in $O(n \log n)$ time. However, a point to note is that while spanners are usually concerned with Euclidean shortest paths (e.g., they have small stretch factors), in our case we use the this k-spanner and its results directly in the approximation algorithm to calculate the approximate cover value.

We use the spanner of visibility graph to compute the approximation algorithm for BCP problem for transparent obstacles. There are relatively less work in literature on non-complete Euclidean graphs, such as, visibility graphs in comparison to the well-studied complete Euclidean graphs. An early result is by Clarkson [29], who showed how to construct a linear sized t-spanner of visibility graph ($t > 1$) in $O(n \log n)$ time without having to construct the visibility graph. This spanner is applied to solve approximate shortest path problem. In [32], Chew shows that the *Constrained Delaunay Triangulation* is a $O(1)$- spanner of the visibility graph and can be constructed in $(O \log n)$ time. Later on, a novel technique of constructing the bounded degree t-spanner of visibility graph in $O(n \log n)$ time is proposed by Das [18], given a collection of polygonal obstacles with n vertices. In our approximation algorithm, we use the concept of the spanner graph [29], precisely named as k-spanner.

2.4 Bellman Ford Shortest Path Algorithm

The Bellman-Ford algorithm [44] solves the single-source shortest paths problem for a graph with both positive and negative edge weights. This algorithm is remarkable in its simplicity, and has the further benefit of detecting whether a negative-weight cycle

is reachable from the source. If there is such a cycle, the algorithm indicates that no solution exists. If there is no such cycle, the algorithm produces the shortest paths and their weights. We use this algorithm to compute the shortest path between the source point s and destination point t of the constructed path in BCP problem for opaque and transparent obstacles. For a planar graph, Bellman-Ford computes the shortest path between a specific source and destination point in $O(n \log n)$ time, n being the number of nodes.

2.5 Summary

We begin this chapter with the related concepts of Coverage and Connectivity in Wireless Sensor Networks. In Section 3.2, we mathematically model sensing, coverage and connectivity. Section 3.3 discuss the related computational geometry data structures and concepts. We briefly explain the Bellman-Ford Algorithm in Section 3.4 which we have used in our shortest path calculation.

CHAPTER 3

RELATED WORK ON COVERAGE PROBLEMS IN WSNs

In this chapter, we investigate the current state of the art research in Coverage problems in Wireless Sensor Network.

Coverage is the measure of QoS of sensing function and is subject to a wide range of interpretations due to large variety of sensors and applications. Considering the coverage concept, different problems can be formulated, based on the subject to be covered (area versus discrete points) and on the design choices, such as sensor deployment method, additional critical requirements, sensing and communication radius as so on; we discuss each of these issues in brief in this chapter. A wide classification can be done with respect to the type of algorithm used as well - centralized versus distributed/localized. We also compare these approaches and algorithms based on their goals, assumptions, complexities and usefulness in practical scenarios. Objectives of these design choices are either to maximize network lifetime; minimize number of sensors; or optimize degree of coverage, and so on. A comprehensive study on coverage connectivity research can be found in [6].

Coverage can be classified of three types based on the subject to be covered. Area Coverage, Point Coverage and Barrier Coverage. The most studied problem is the area coverage where the main objective of the sensor network is to cover(monitor) an area. Research is going on in both the static and mobile sensor network [20, 3].

The design choices can be stated as:

1. Sensor Deployment Strategies: deterministic versus random. A deterministic sensor placement may be feasible in friendly and accessible environments. Random sensor distribution is generally considered in military applications and for remote or inhospitable areas.

2. Energy Requirements: In the most typical scenarios, energy requirement is a big factor as sensors are usually limited with respect to its battery life. Several research work has been done on energy efficient coverage.
3. Sensing and Communication Radii: Homogeneous/ Heterogeneous sensor network is the subject of interest here. While constraints are less in homogeneous sensor network; heterogeneous sensor network has a wider scope in applications.

A broader classification of coverage problems can also be done in terms of their goals, assumptions, algorithm complexities and practical applicability. The three categories are:

1. Coverage based on the exposure path
2. Coverage based on sensor deployment strategies
3. Miscellaneous Strategies

Next, we illustrate each of these three categories in more detail.

3.1 Coverage Based On Exposure Path

Coverage based on the exposure path in WSNs is essentially a combinatorial optimization problem. Two kind of coverage is considered in this case. One is to compute *Best Coverage* [13, 46, 38, 21] and another is to compute *Worst Coverage* path [13, 38, 21].In the Worst Coverage Path, the problem is to find that path in the sensor field that has minimum observability. Hence, the probability of detecting the moving object would be minimum. This is important in real application scenario as this can be the preferred path of adversaries; hence knowing this path helps to add new nodes in the sensor field and hence observability increases. Two well known methods of for the worst case coverage problem are Minimal Exposure Path [13] and Maximal Breach Path [13, 46] .

On the other hand, in the best case coverage, the goal is to find a path that has the highest observability, and hence an object moving along that path will be most probable to be detected by the nodes [13, 46]. This is particularly important into those applications where security is of highest concern. Two approaches have been used to

solve this problem, *Maximal Exposure Path* [13] and *Maximal Support Path* [21]. In the following subsections, we discuss several methods to calculate the worst case and best case coverage paths and the algorithms that use the concept of exposure to derive analytical results.

Another important work, investigated by Liu and Towsley [8], is to determine the number of sensor nodes to be randomly deployed in the field such that the probability of a penetration path is close to zero. In this exposure based model, sensing abilities of the sensors diminish as the distance increases, but another important factor is the sensing time (exposure). This modeling is done considering sensing time, intensity of the sensor field, and so on

3.1.1 Minimal Exposure Path : Worst Case Coverage

Coverage is a measure of how well a sensing field is covered with sensors. Informally stated, it can be defined as the expected average ability of observing a target moving in the sensing field. The Minimal Exposure Path provides valuable information about the worst case coverage in sensor networks.

The basis of the proof adopted to compute the exposure path of one sensor in [39] lies in the fact that since any point on the dotted curve is closer to the sensor than any point lying on the straight line segment along the edge of the square, the exposure is more in the former case. Also, since the length of the dotted curve is longer than the line segment, the dotted curve would induce more exposure when an object travels along it, given that the time duration is the same in both the cases. Furthermore, this method is extended when the sensing region is a convex polygon and the sensor is located at the center of that inscribed circle.

This intuition can further be extended to compute the minimal exposure path under the scenario of many sensors. To simplify, the problem can be transformed from the continuous domain into a tractable discrete domain by using an $m \times n$ grid [39]. The minimal exposure path is then restricted to straight line segments connecting any

two consecutive vertices of a grid square. This approach transforms the grid into an edge weighted graph and computes minimal exposure path using Dijkstras Single Source Shortest Path Algorithm.

3.1.2 Maximal Exposure Path : Best Case Coverage

A maximal exposure path between two arbitrary points s and t in a sensing field is a path following which the total exposure is maximum. It can be interpreted as a path having the best case coverage. It has been proved in [21] that finding the maximal exposure path is NP-hard because it is equivalent to finding the longest path in an undirected weighted graph, which is known to be NP-hard. However, there exist several heuristics to achieve near-optimal solutions under the constraints that the objects speed, path length, exposure value and times:

- Random Path Heuristics: In this method, a random path is created according to the rule that a node on the shortest path from source to destination is selected at certain times and a random node is selected at other times. Nodes on the shortest path are selected because of the time constraint and random nodes are selected to collect more exposure..
- Shortest Path Heuristic: In this approach, first a shortest path is calculated between the two end points assuming certain topographical knowledge is available.
- Longest Path Heuristics: This heuristic superimposes a grid over the exposure path and then finds the shortest path to each grid point from s and t.
- Adjusted Best Point Heuristics: This method improves the best-point heuristic by considering paths that consist of multiple shortest paths.

3.1.3 Maximal Breach Path : Worst Case Coverage

As discussed in Section 3.1.1 that finding a minimal exposure path is equivalent of finding a worst case coverage path, which provides valuable information about node deployment density in the sensing field. A very similar concept to find out worst case

coverage paths is the notion of maximal breach paths [13]. A maximal breach path through a sensing field starting at s and ending at t is a path, such that for any point p on the path, the distance from p to the closest sensor is maximum. The concept of Voronoi diagram [33], a well known construct from computational geometry is used to find a maximal breach path in a sensing field. It is also proved intuitively since by construction, the line segments in a Voronoi diagram maximizes the distance from the closest sites, the maximal breach path must lie along the Voronoi edges. The algorithm then checks the existence of a path from s to t using breadth-first-search (BFS) and then uses binary search between the smallest and largest edge weights in the computed Voronoi graph to find the maximal breach path.

3.1.4 Maximal Support Path: Best case coverage

A maximal support path through a sensing field starting at s and ending at t is a path, such that for any point p on the path, the distance from p to the closest sensor is minimized. This is similar to the concept of maximal exposure path. However, the difference lies in the fact that a maximal support path algorithm finds a path at any given time instant, such that the exposure on the path is no less than some particular value which should be maximized. A maximal support path in a sensing field can be found by replacing the Voronoi diagram by its dual, Delaunay triangulation where the edges of the underlying graph are assigned weights equal to the length of the corresponding line segments in the Delaunay triangulation [13, 46].

This ends our brief discussion on coverage problems based on exposure paths in WSNs. Next, we discuss different deployment strategies which impact coverage in WSNs.

3.2 Coverage Based On Sensor Deployment Strategies

The second approach to the coverage problem is to find sensor deployment strategies that would maximize coverage as well as maintain a globally connected network graph. Several deployment strategies have been studied for achieving an optimal sensor

network architecture that would minimize cost, provide high sensing coverage, and be resilient to random node failures etc. The most usual deployment strategy of sensor nodes are random deployment. However, random placement does not guarantee full coverage because it is stochastic in nature, hence often resulting in accumulation of nodes at certain areas in the sensing field whereas leaving other areas deprived of nodes. Keeping this in mind, some of the deployment algorithms try to find new optimal sensor locations after an initial random placement and moves the sensors to those locations, achieving maximum coverage. These algorithms are applicable to only mobile sensor networks. Research has also been conducted in mixed sensor networks, where some of the nodes are mobile and some are static; and approaches are proposed to detect coverage holes after an initial deployment and trying to heal or eliminate those holes by moving sensors. It should be noted that an optimal deployment strategy should result not only in a configuration that would provide sufficient coverage, but also satisfy certain constraints such as node connectivity and network connectivity [40].

Next, in the following subsections, we discuss several research efforts on this category in brief.

3.2.1 Imprecise Detection Algorithm (IDA)

In [41], an effective sensor nodes placement strategy problem is solved so that it can be covered effectively for surveillance and target detection. This algorithm is known as Imprecise Detection Algorithm(IDA); IDA is grid-based coverage algorithm, proposed to ensure that every grid point is covered with a minimum confidence level. IDA considers a minimalistic view of a sensor network by deploying a minimum number of sensors on a grid that would transmit a minimum amount of data.

The algorithm as described in [41] takes three inputs: M, $M*$, M_{min}; where M is the miss probability matrix as mentioned above, $M* = (M_1, M_2, \ldots, M_N)$ such that M_i is the probability that a grid point i is not collectively covered by the set of sensors, $M_{min} = 1 - T$ is the maximum value of the miss probability that is permitted for any

grid point. The algorithm is iterative and uses a greedy heuristic to determine the best placement of one sensor at a time. It terminates either when a preset upper limit on the number of sensors is reached, or sufficient coverage of the grid points is achieved. The time complexity of the algorithm is $O(n^2)$, where n is the total number of grid points in the sensor field. It attempts to evaluate the global impact of an additional sensor by summing up the changes in the miss probabilities for the individual grid points.

3.2.2 Potential Field Algorithm (PFA)

In [3, 40], a potential field based deployment approach using mobile autonomous robots is proposed to maximize the area coverage. This is known as Potential Field Algorithm(PFA). In [40], the scheme is augmented such that each node has at least k neighbors. The potential field technique using mobile robots was first introduced in [35].

The basic concept of potential field is that each node is subjected to a force F $(vector)^2$ that is the gradient of a scalar potential field U. That is, $F = - \bigtriangledown U$. Each node is subjected to two kinds of forces: a) F_{cover} that causes the nodes to repel each other to increase their coverage and b) F_{degree} that constrains the degree of nodes by making them attract towards each other when they are on the verge of being disconnected. The forces are modeled as inversely proportional to the square of the distance between a pair of nodes and they obey the two boundary conditions.

3.2.3 Virtual Force Algorithm (VFA)

Similar to the potential field approach as described in [40], a sensor deployment algorithm based on virtual forces is proposed in [[49, 50] to increase the coverage after an initial random deployment. Since a random placement does not guarantee effective coverage, an approach that modifies the sensor locations after a random placement is useful.

A sensor is subjected to three kinds of forces, which are either attractive or repulsive in nature. In the VFA model, obstacles exert repulsive forces (F_{iR}), areas of preferential

coverage (sensitive areas where a high degree of coverage is required) exert attractive forces (F_{iA}), and other sensors exert attractive or repulsive forces (F_{ij}), depending on the distance and orientation. A threshold distance, d_{th}, is defined between two sensors to control how close they can get to each other. Likewise, a threshold coverage c_{th} is defined for all grid points such that the probability that a target at any grid point is reported as being detected is greater than this threshold value.

For a $n \times m$ grid with a total number of k sensors deployed, the computational complexity of the VFA algorithm is $O(nmk)$. Negligible computation time and a one-time repositioning of sensors are two of its primary advantages. However, the algorithm does not provide any route plan for repositioning the sensors to avoid collision.

3.2.4 Integer Linear Programming Algorithm (ILPA)

Chakrabarty et. al in [49, 50], model the optimization problem of coverage with Integer Linear Programming (ILP) and represent the sensor field as a two or three dimensional grid. Given a variety of sensors with different ranges and costs, they provide strategies for minimizing the cost, provide coding-theoretic bounds on the number of sensors and present methods for their placement with desired coverage. Their approach of maximizing coverage in the sensing field is different in the sense that it determines a deployment strategy, such that every grid point is covered by a unique subset of sensors. In this way, the set of sensors reporting a target at a particular time uniquely identifies the grid location for the target at that time.

Some more potential works in this field are *Distributed Self Spreading Algorithm*, *VEC, VOR and MiniMax Algorithm*, *Bidding Protocol Incremental Self Deployment Algorithm* and so on. The book chapter [6] has a comprehensive literature survey of these problems. Next, we discuss the third category, the miscellaneous strategies of coverage research in WSNs.

3.3 Miscellaneous Strategies

Our discussion so far, has concerned mainly with algorithms that guarantee optimal coverage of the sensing field. However, as mentioned earlier, a sensor network needs to be connected as well, so that the data sensed by the nodes can be transmitted by multi-hop communication paths to other nodes and possibly to a base station where intelligent decisions can be made. Therefore, it is equally important for a coverage algorithm to ensure a connected network. In this section, we discuss a few techniques that ensure coverage as well as connectivity in a sensing field, while at the same time reduces redundancy and increases overall network life time.

It is envisioned that a typical wireless sensor network would consist of large numbers of energy-constrained nodes deployed with high density. In such a network, it is sometimes undesirable to have all the nodes to be in the active state simultaneously. In addition, keeping all the nodes active simultaneously would dissipate energy at a much faster rate and would reduce overall system lifetime. Hence, it is important to turn off the redundant nodes and maximize the time interval of continuous-monitoring, transmitting or receiving function. Scheduling of nodes that would control the density of active nodes in a sensor network has been the focus of many research works.

One of the important design aspects in WSNs is to find the minimum number of sensor nodes which can cover a given domain with a certain degree of precision while maintaining the global connectivity in the network. Finding the minimum number of nodes for the connected coverage has been identified as an NP hard problem [22, 5]. Therefore existing solutions [5] give different heuristics to optimize it.These problems [22, 5], related to optimizing number of sensor nodes have got two variations - connected coverage for homogeneous sensor network [22] and connected coverage for heterogeneous sensor network [51, 52]. A related research problem is solved in [51] to find the optimum number of sensor nodes in a fault tolerant heterogeneous sensor network. Furthermore, [42] tries to find an optimum scheduling strategy so that the network becomes resilient to

power consumption and in turn gives a longer life time. In addition to that, Integrated coverage and connectivity problem is studied in [48].

An energy efficient node scheduling based coverage mechanism is proposed in [14]. The protocol proposed is distributed and localized. The off duty eligibility rule determines whether a node's sensing area is included in its neighbors' sensing area. This algorithm described in [14] consists of two phases: self scheduling phase and sensing phase. In the self scheduling phase, each sensor broadcasts its position and node id, and listens to the advertisement messages from its neighbors to obtain their location information. Then it calculates the sponsored sectors by its neighbors and checks whether the union of their sponsored sectors can cover its own sensing area. If so, it decides to turn itself off. However, if all the nodes make decisions simultaneously, blind spots might appear. To avoid such a situation, each node waits a random period of time and also broadcasts its status message to other nodes. In this way the nodes self-schedule, thus reducing energy consumption while maintaining the original coverage area.

This ends our discussion on related work on coverage problems in WSNs based upon these three categories.

A different classification can also be done in terms of the varied solution techniques of these problems. In general, to solve coverage related problems, both deterministic and probabilistic models have been used. Existing solutions such as [3, 40, 41] use the probabilistic model and geometric random graph theory and graph algorithms; as the structure of Geometric Random Graphs (GRG) provides the closest resemblance to wireless sensor networks. Researches are there on coverage models for Energy Efficient Random Coverage, Connected random Coverage, Deterministic Coverage, Node Coverage Approximation etc. While the first two are the randomized model, the third one is deterministic. Solutions for determining whether a node's coverage can be sponsored by its neighbors (sponsored coverage calculation) is solved in [14]. These solutions are intended for Energy Efficient Random Coverage. The solution there can also be random

or deterministic. On the contrary, Computational geometry and different graph search algorithms are used in existing solutions such as [13, 46, 38, 21] etc.

In terms of mobile sensor network, remarkable research is going on the coverage and connectivity there as well. In contrast to static sensor networks, nodes in mobile sensor networks are capable of moving in the sensing field. Such networks are capable of self deployment starting from an initial configuration. The nodes spread out such that coverage in the sensing field is maximized while maintaining network connectivity.

However, there are significant differences between the problem we consider and these other works. To the best of our knowledge, ours is one of the first efforts to study the presence of obstacles in coverage problems in WSNs.

3.4 Summary

Chapter 3 is the literature study on coverage problems in Wireless Sensor Networks. We discuss existing research works in terms of their goals, assumptions, algorithm complexities and practical applicability.

CHAPTER 4

$BCP(s,t)$ PROBLEM FOR OPAQUE OBSTACLES

In this chapter we develop an algorithm for the $BCP(s,t)$ problem for opaque obstacles which we have defined in Chapter 1. We also analyze the running time of the algorithm and prove its correctness. In this chapter, we show, that obstacles pose significant challenges in the computation of $BCP(s,t)$. Existing techniques of solving it by creating the Delaunay Triangulations of sensor nodes or its sparse subgraphs such as, GG or RNG can not be used here. Even visibility graph, a proximity based data structure used in numerous obstacle avoiding shortest path computation problems in computational geometry is also not adequate for $BCP(s,t)$ for the opaque obstacles.

Let us first discuss why the presence of obstacles make the best coverage problem difficult. As discussed earlier, the visibility graph, a standard data structure used for numerous proximity problems in the presence of obstacles, does not necessarily contain the BCP (Recall Figure 1.2 which clearly shows that the BCP from s to t, shown as a dotted path, is not contained in the constructed visibility graph of the 2 sensors and 4 opaque obstacles). Besides the shortcomings of the visibility graph, the existing solutions for the best coverage path problem without obstacles [13, 46, 38] depend on structures such as the Delaunay triangulation, Gabriel graph, relative neighborhood graph and so on. These structures have no easy generalizations to the case of obstacles.

4.1 Outline of the Algorithm $BCP(s,t)$ for Opaque Obstacles

We construct the constrained Voronoi diagram of n sensor sites in presence of m line obstacles (where each of the sites have the same weight, i.e., 1; that is why we omit the term weighted in this work). Next we construct a specific *dual* graph of this Voronoi diagram, such that the best coverage path is guaranteed to be contained in this dual graph. We assign edge weights to each of the constructed edges in the dual graph,

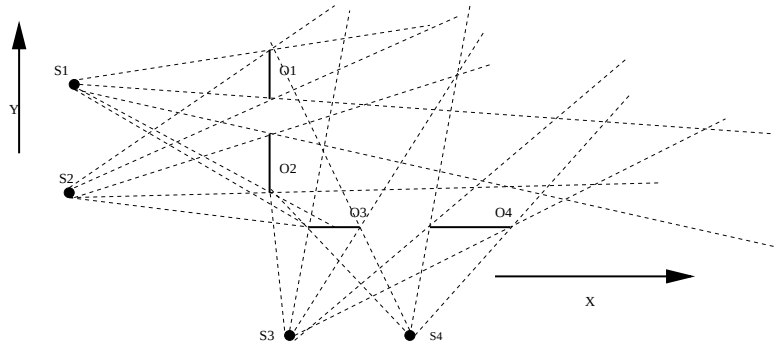

Figure 4.1. A worst case lower bound of the Constrained Voronoi Diagram.

where the weight of each edge is the distance from its least protected point to its nearest sensor. The dual creation and edge weighing is defined in the Subsection 4.1.1. Finally, we compute path between points s and t in the constructed weighed dual graph whose largest edge is smaller than the largest of any other path in the graph. This computation is accomplished using the Bellman-Ford algorithm [44].

As mentioned in Chapter 2, the constrained Voronoi diagram has $\Omega((mn)^2)$ combinatorial complexity, where n is the number of sensor sites and m is the number of line obstacles. We informally describe why this is so. Consider Figure 4.1 depicting the Voronoi diagram of 4 sensors and 4 line segment obstacles. The sensors are S_1, S_2, S_3 and S_4. Obstacles are O_1, O_2, O_3 and O_4. Generalizing for n sensors and m line obstacles, each of the vertically placed $O(n)$ sensors see through $O(m)$ "gaps" between the vertically placed obstacles, creating a combinatorial structure of $O(nm)$ *arrangement* of lines. A similar combinatorial structure is also created by the horizontally placed sensors. The overlay of these two combinatorial structures will give rise to an $O(n^2m^2)$ combinatorial structure, where each cell is a Voronoi cell whose interior points are closest to a specific sensor. (This arrangement becomes more complicated when we have to take into account the perpendicular bisectors between every pair of sensors).

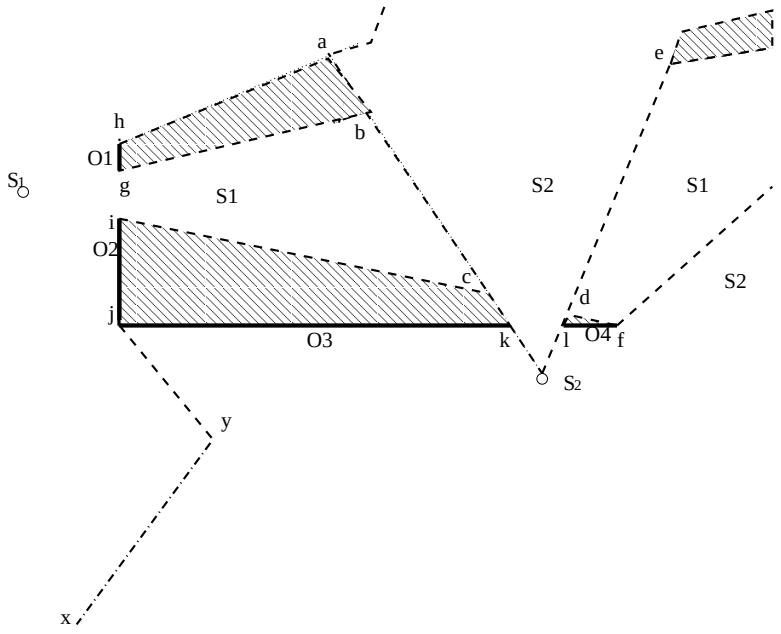

Figure 4.2. Constrained Voronoi diagram of sensors and obstacles.

4.1.1 Dual of Constrained Voronoi Diagram

In this section we discuss how to define a suitable dual graph of the constrained Voronoi diagram. As an example, consider Figure 4.2 which shows the Voronoi diagram where S_1, S_2 are two sensors and O_1, O_2, O_3, O_4 are four obstacles. The filled areas are dark regions which cannot be sensed by either of the two sensors S_1, S_2. The remaining cells of the Voronoi diagram are labeled by the sensors to which they the closest. Notice that unlike Voronoi diagrams without obstacles, the Voronoi region associated with a sensor may be disconnected, consisting of several cells.

We define the *dual* of the constrained Voronoi diagram.

4.1.1.1 Dual Definition

The dual of the Constrained Voronoi diagram (C-Voronoi Diagram) is a weighted graph. The vertices are the union of (a) the set of sensors, (b) the endpoints of the obstacles, (c) the vertices of the Voronoi diagram not already included in the first two sets (i.e., the third type of vertices are intersections of perpendicular bisectors between sensors), and (d) the points s and t. We next define the edges of the dual graph.

Consider any edge $e = (u, v)$ of the C-Voronoi diagram. The edge is one of three types: (a) it is part of an obstacle, (b) it is part of a perpendicular bisector between two sensors, or (c) it is part of an extension of a visibility line from a sensor that passes through an obstacle endpoint. For example, in Figure 4.2, edge (g, b) is of type (a), edge (x, y) is of type (b), and edge (b, c) is of type (c). Let $C_1(e)$, $C_2(e)$ be two adjacent C-Voronoi cells on either side of the edge. Assume that neither of the cells are dark, and let S_1 and S_2 be the labels on these cells. Note that if e is of type (b), then e and one of the two sensors are collinear (e.g., in Figure 4.2 the edge (b, c) is collinear with sensor S_2).

For each such edge $e = (u, v)$ of the C-Voronoi diagram, we add edges to the dual graph as follows:

- We add four dual edges (u, S_1), (u, S_2), (v, S_1), and (v, S_2) (as shown in Figure 4.3). Each dual edge is assigned a weight equal to the Euclidean distance between its endpoints.

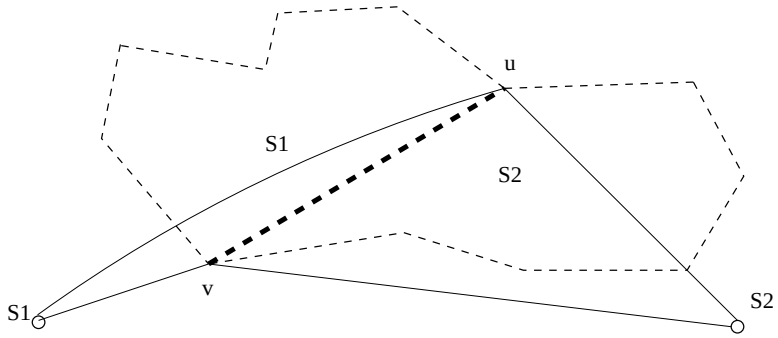

Figure 4.3. The four dual edges corresponding to the Voronoi edge $e(u, v)$.

- In addition, if the edge e is of type (b) i.e., it is part of a perpendicular bisector between S_1 and S_2, and *such that* the S_1 and S_2 can see each other and the line connecting S_1 to S_2 passes through e, then we place an additional dual edge $(S_1(e)S_2(e))$. This "direct" dual edge gets weight equal to half the Euclidean dis-

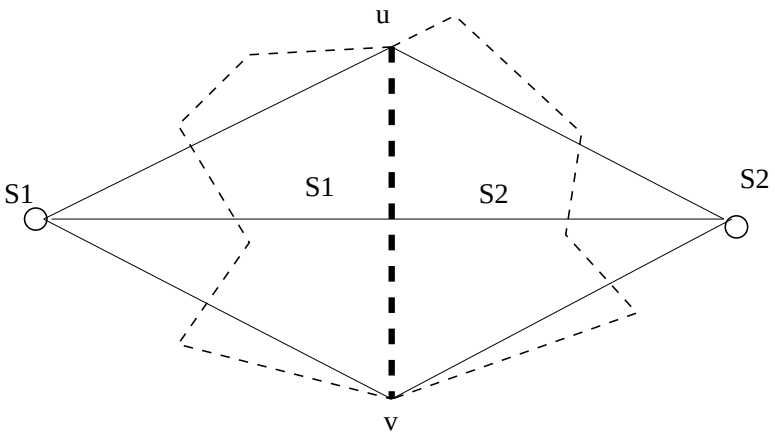

Figure 4.4. The additional fifth "direct" dual edge S1S2.

tance between S_1 and S_2. Figure 4.4 shows an example of this additional "direct" dual edge.

Next, for each edge of the C-Voronoi diagram such that one of the adjacent cells is dark, we place two edges between the sensor associated with the other cell and the endpoints of the Voronoi edge. Each such dual edge gets weight equal to their Euclidean distance. Finally, we connect s and t to their closest visible sensors (assuming at least one sensor can see them), and weigh these dual edges by their Euclidean distances. This concludes the construction of the dual graph.

Lemma 1 *The dual graph has $O(m^2n^2 + n^4)$ number of vertices and edges.*

Proof 1: As the Voronoi diagram has $O(m^2n^2+n^4)$ number of vertices and edges [9], so has the dual, as per the dual definition. Since each C-Voronoi edge contributes a constant number of edges to the dual, the number of edges in the dual is also $O(m^2n^2 + n^4)$.

We note that way the dual graph has been constructed, at least one endpoint of each edge is a sensor. Thus there cannot be two or more consecutive vertices along any path that are not sensors. We shall next show that there exists a $BCP(s, t)$ that has such a property, hence it can be searched for within this dual graph.

4.2 The $BCP(s,t)$ Algorithm for Opaque Obstacles

The algorithm to compute $BCP(s,t)$ for opaque obstacles is as follows::

Algorithm 1 Calculate $BCP(S,O,s,t)$ for Opaque Obstacles

1: Using the technique of [9], construct the constrained Voronoi diagram of all n sensors and m obstacles (assign each sensor a weight of 1).
2: Construct the dual of this C-Voronoi diagram as described in Section 4.1.1.
3: Run *Bellman-Ford* algorithm on this constructed dual graph starting at point s and ending at point t, which computes the *Best Coverage Path* between s and t.
4: The value of $Cover = \max(weight(e_1), weight(e_2), \ldots, weight(e_r))$ in the constructed path, where $e_1, e_2, \ldots, e_r$ are the edges in the best coverage path, $BCP(s,t)$.

4.2.1 Proof Of Correctness

Theorem 1 *An $BCP(s,t)$ path for opaque obstacles is contained within the constructed dual graph.*

Proof 2: The overall idea of the proof is to show that a best coverage path that lies outside the dual graph can be "transformed" into one that only uses the edges of the dual graph. Consider Figure 4.5, which shows a best coverage path that does not use the edges of the dual graph. Let us decompose this path into pieces such that each piece lies wholly within a cell of the Voronoi diagram. Consider one such piece within a cell labeled S_i. Let the piece start at a point p and end at a point q, where both p and q are along the cell's boundary. It is easy to see that each such piece can be replaced by the two line segments (p, S_i) and (S_i, q) *without* increasing the cover of the path. Thus, any best coverage path can be transformed into one having linear segments that goes from cell boundary to sensor to cell boundary and so on.

We next show that the points along the cell boundaries of this transformed path are actually Voronoi vertices. To prove this, assume that one such point is not a Voronoi vertex, i.e, it is a point along an edge of a cell boundary. Consider Figure 4.6 which

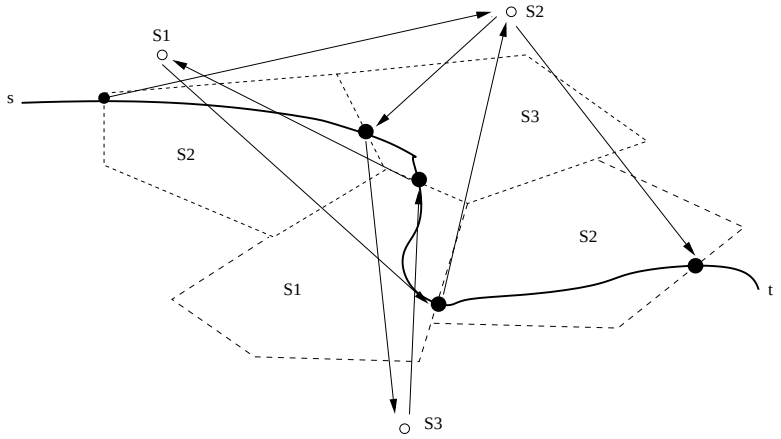

Figure 4.5. Transforming a best coverage path.

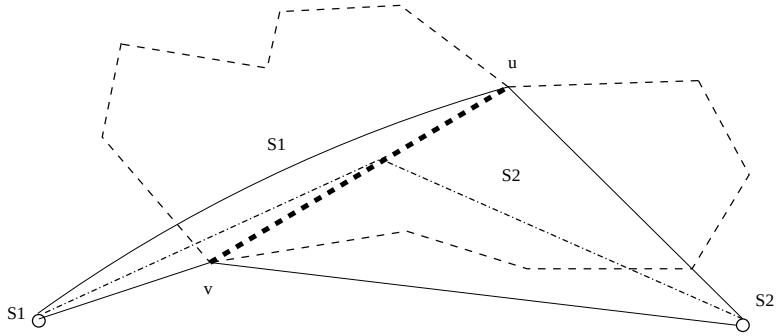

Figure 4.6. Moving a $BCP(s,t)$ vertex to a Voronoi vertex.

shows a portion of the $BCP(s,t)$ (transformed as discussed above) that goes from sensor S_1 to a point along the Voronoi edge (u,v) and then to sensor S_2. It should be clear that if we replace this portion by two edges of the dual graph, (S_1,v) and (v,S_2), we will achieve an alternate path whose overall cover value will be the same or less.

One final case needs to be discussed. If the $BCP(s,t)$ passes through a point along a Voronoi edge that is a perpendicular bisector between two sensors S_1 and S_2 that such that there is a "direct" dual edge between S_1 and S_2, then the portion of $BCP(s,t)$ from S_1 to S_2 can be replaced by this direct edge without increasing the overall cover value of the path. This situation is shown in Figure 4.7.

Thus we conclude, that, for opaque obstacles there exists an $BCP(s,t)$ that only follows the edges of the dual graph.

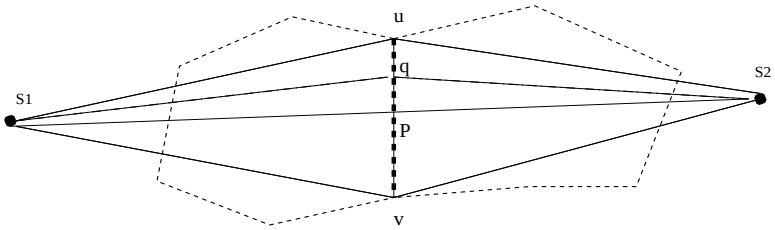

Figure 4.7. Moving a $BCP(s,t)$ vertex to the middle of the dual edge.

4.2.2 Running Time Analysis

Using the techniques in [9], Step 1 of our algorithm can be accomplished in $O(m^2n^2 + n^4)$ time and space. Likewise, constructing the dual is straightforward, as we have to scan each edge of the Voronoi diagram and insert the corresponding dual edges with appropriate weights. This also takes $O(m^2n^2 + n^4)$ time. Finally running the Bellman-Ford Algorithm on a graph with $O(m^2n^2 + n^4)$ number of vertices and edges takes $O((m^2n^2 + n^4)\log(mn + n^2)))$ time.

4.3 Summary

This chapter illustrated the intuition for solving $BCP(s,t)$ problem for opaque obstacles. We depicted the challenges of modeling opaque obstacles in computation of best coverage path. Next, we outline the algorithm. In addition to that, we explain the concept of the created dual graph of the constrained Voronoi diagram and describe the weight assignment policy of the dual graph edges. In continuation, we state the algorithm and prove its correctness. We conclude this Chapter with complexity analysis of $BCP(s,t)$ algorithm for opaque obstacles.

CHAPTER 5

$BCP(s,t)$ PROBLEM FOR TRANSPARENT OBSTACLES

In this Chapter, we study the $BCP(s,t)$ problem for transparent obstacles. Note that for transparent obstacles, computing $BCP(s,t)$ is an easier problem than that for opaque obstacles. In contrast to the BCP for opaque obstacles, in the case of transparent obstacles we can show that the visibility graph contains the BCP. However, unlike traditional visibility graph, the edge weights of this graph are more complex than standard Euclidean distances. Consequently, the running time of the algorithm is dominated by the edge weighing task.

We propose two algorithms for the $BCP(s,t)$ problem for transparent obstacles: an exact algorithm and an approximation algorithm. We prove the correctness and running time analysis for both algorithms (as well as an approximation bound for the approximation algorithm). Similar to the opaque case, here also both algorithms first create a weighted graph and then search for the best coverage path within this graph using the Bellman-Ford algorithm.

The exact algorithm uses a quadratic-size visibility graph whereas the approximation algorithm uses a linear-sized k-spanner of visibility graph. The linear size of the spanner graph makes the approximate algorithm more efficient than the exact one in terms of time complexity.

5.1 Exact $BCP(s,t)$ Algorithm for Transparent Obstacles

As mentioned, computing $BCP(s,t)$ is relatively an easier problem than that of opaque obstacles: as here obstacles do not block sensing, so the complication that is described in Figure 1.2 does not arise here . In fact, as we shall see, the best coverage path indeed follows the edges of the visibility graph.

The outline of the algorithm is as follows. We create the visibility graph of n sensors and m line segment obstacles, as well as the points s and t. The graph has $O(m^2 + n^2)$ edges. We then weigh each of the edges of the constructed visibility graph in a specific manner. Figure 5.1 explains the edge weighing process. A normal Voronoi diagram of the n sensor points (i.e., ignoring obstacles) is first overlayed on top of the constructed visibility graph. Consider the example in Figure 5.1, where the visibility edge (u, v) has to be weighed. The edge (u, v) passes through the Voronoi cells of sensors u, m, n and v. Let us partition (u, v) into segments that lie wholly within these Voronoi cells. For each segment lying inside the Voronoi cell of a particular sensor, we find out the least protected point. This has to be either of the two endpoints of this segment - For example, (u, v) intersects the Voronoi cell of m at the points a and b, so either a or b is the least protected point for this segment, and the cover value of the segment is $\max(||am||, ||bm||)$. We compute the cover value of all such segments that belong to (u, v), and the maximum of these values is the cover value of (u, v), which gets assigned as the weight of (u, v). Once this weighted visibility graph has been constructed, we run the Bellman-Ford algorithm [44] to compute the best coverage path between s and t. The algorithm is formally described in Algorithm-2.

Algorithm 2 Calculate Exact $BCP(S, O, s, t)$ for Transparent Obstacles

1: Construct the visibility graph of n sensor nodes and m line obstacles, the starting point s and end point t. Let E be the edge set of the visibility graph.
2: Construct the (normal) Voronoi diagram of the n sensor nodes.
3: **for** each edge $(u, v) \in E$ **do**
4: Partition (u, v) into segments $(u = a_1, b_1), (a_2, b_2), \ldots, (a_{r'}, b_{r'} = v)$ where each segment (a_i, b_i) lies wholly within a Voronoi cell of sensor S'_i, say $Cell(S'_i)$.
5: **for** each segment (a_i, b_i) **do**
6: Find the least protected point of that segment, which is either a_i or b_i.
7: $maxpiece_i = \max(||(S'_i a_i||, ||S'_i b_i||)$
8: **end for**
9: $weight((u, v)) = \max(maxpiece_1, \ldots, maxpiece_{r'})$,
10: **end for**
11: Run the Bellman-Ford algorithm on this weighted visibility graph starting at point s and ending at point t to compute the $BCP(s, t)$.
12: For the computed $BCP(s, t)$ path, the value of $Cover = \max(weight(e_1), weight(e_2), \ldots, weight(e_r))$, where $e_1, e_2, \ldots, e_r$ are the edges of the best coverage path, $BCP(s, t)$.

Theorem 2 *An exact $BCP(s, t)$ for transparent obstacles is contained within the constructed visibility graph.*

Proof 3: As in the proof of Theorem 1, the idea here is to show that a best coverage path that lies outside the visibility graph can be "transformed" into one that only uses the visibility edges. A best coverage path that does not follow the visibility edges means that the path makes some bend either:

- *Case 1:* Inside a Voronoi cell, or
- *Case 2:* At a Voronoi bisector, or

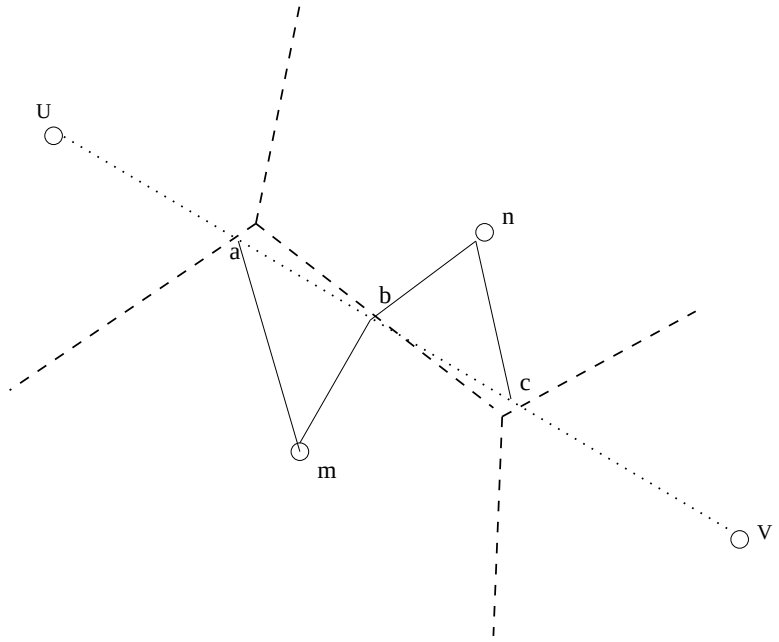

Figure 5.1. Weighing of a visibility edge uv.

- *Case 3:* At a Voronoi vertex

We describe the transformations necessary for bends of the type Case 1. Consider Figure 5.2, which shows a best coverage path from s to t that crosses through the Voronoi cell of sensor S_i but that does not follow visibility graph edges (the line obstacles are shown as solid thick lines, and the best coverage path is shown as a wriggly path).

Consider two points a and b along this path inside the cell. Clearly the cover value of the portion of the path from a to b is at least $\max ||S_i a||, ||S_i, b||$. Thus if we replace the portion from a to b by the straight line segment (a, b), it is easy to see that the cover value of the transformed path will not have increased.

Thus, if we apply this "tightening" operations to the best coverage path, we shall be able to eliminate all bends of the type described in Case 1. The resulting portion of the best coverage path within the Voronoi cell for S_i will be eventually transformed to as shown in Figure 5.2. As can be seen, other than the two points on the boundary of the cell (i.e., vertices of type Case 2 or Case 3), the rest of the vertices of the path in the interior of the cell will be obstacle endpoints.

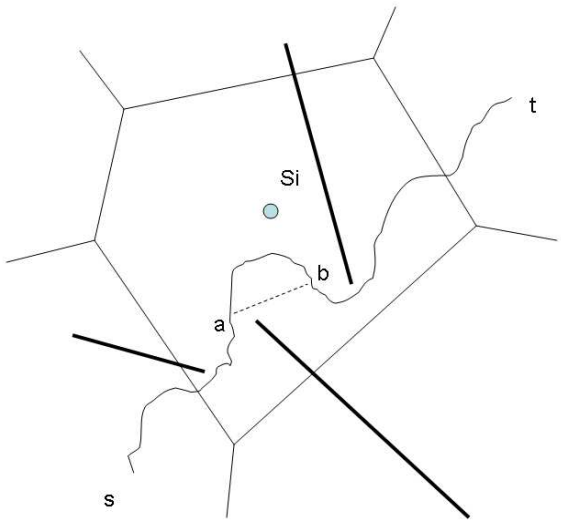

Figure 5.2. Eliminating bends within Voronoi cells.

If we apply the above transformation to all Voronoi cells, we can eliminate all vertices of type Case 1 from the path.

Next, we illustrate the case 2 type bend elimination with Figure 5.4. As shown in the Figure, Consider two points a and b along this on the Voronoi bisector. Clearly the cover value of the path from S_i to S_j is at least $1/2||S_iS_j||$. Thus, if we replace the portion from a to b by the straight line segment (a, b), it is easy to see that the cover value of the transformed path will not have increased.

Thus, if we apply this "tightening" operations to the best coverage path on the bisector, we shall be able to eliminate all bends of the type described in Case 2. The resulting portion of the best coverage path from Voronoi cell S_i to S_j will be eventually transformed to as shown in Figure 5.5. So, with elimination of case 1 and case 2 bends, the other bends would only be those which bends at Voronoi vertex.

The elimination of vertices of type Case 3 follow similar arguments and we omit discussing them in this work. Thus, we conclude that the visibility graph contains a best coverage path.

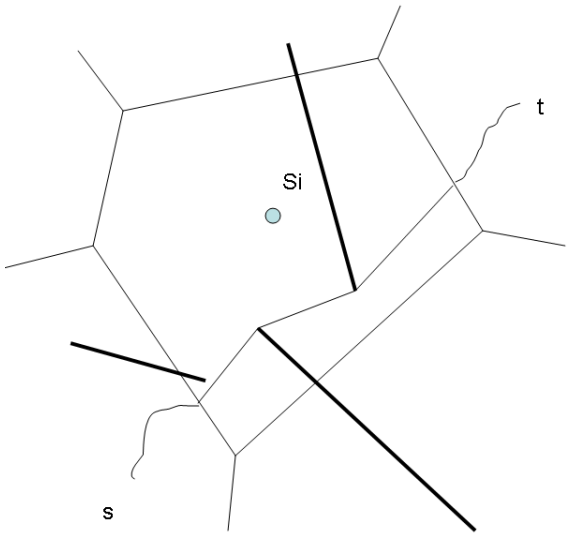

Figure 5.3. "Tightening" paths within Voronoi cells.

We now analyze the running time of the algorithm.

Lemma 2 *The running time of the Exact $BCP(s,t)$ algorithm for transparent obstacles is $O(nm^2 + n^3)$.*

Proof 4: There exists *output sensitive* algorithm [37] to compute the visibility graph. In our case the running time will be $O((m+n)\log(m+n)+x)$, where x is the number of edges in visibility graph. Constructing the Voronoi diagram of n nodes can also be done in $O(n \log n)$ time. Assigning weights to each visibility edges take $O(n)$ times, as we have to partition each edge into pieces that lie wholly within a cell and compute the covers of each piece. So, assigning edge weights to all $O(m^2+n^2)$ edges takes $O(nm^2+n^3)$ time. Running the Bellman-Ford algorithm will take $O((m+n)\log(m+n))$ time. So, overall the running time of this algorithm is dominated by weight assignment of all edges.

5.2 Approximated $BCP(s,t)$ Algorithm for Transparent Obstacles

We design an approximation algorithm for the $BCP(s,t)$ problem for transparent obstacles using the k-spanner of the visibility graph [29]. The outline of this algorithm is exactly same as the exact $BCP(s,t)$ algorithm for transparent obstacles, with the change

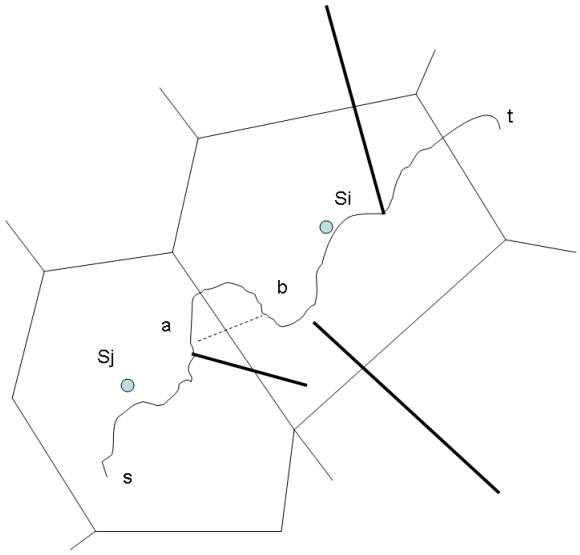

Figure 5.4. Eliminating bends on the bisector.

only in the use of the spanner graph rather than the visibility graph. The algorithm is described below.

Algorithm 3 Calculate Approximated BCP(S, O, s, t) for transparent obstacles

1: **Construct** the *bounded degree k spanner of Visibility Graph* of n sensor nodes and m line obstacles. For that, we follow the same strategy as it is been done on the [18].The graph has linear number of edges and is $O(m+n)$.

2: **Repeat** all $2-12$ steps of Algorithm 2.

Theorem 3 *The approximation factor of the algorithm Approximated* $BCP(s,t)$ *for transparent obstacle is* $O(nk)$.

Proof 5: Let us explain the notion of spanner with the help of figure 5.6. According to the definition, the distance from vertices u to v in a spanner is at most k times the distance from u to v in the visibility graph (where k is a constant, known as the *stretch factor* of the spanner).

The worst case arises when all n sensors have the same influence on the coverage of a single line. In Figure 5.7, we consider that case, where

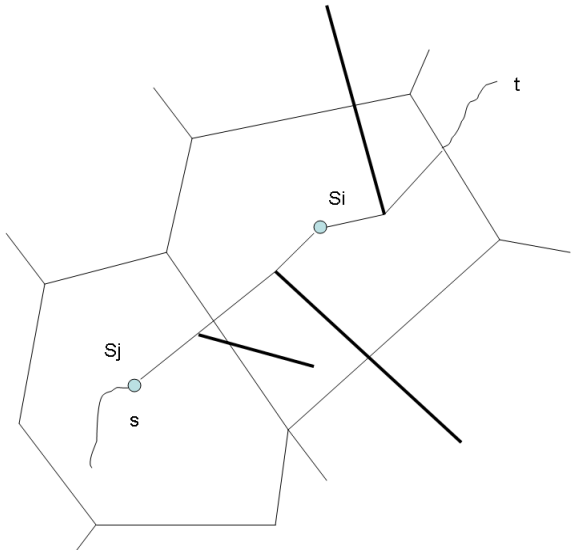

Figure 5.5. "Tightening" paths on the bisector.

- u and v are two points and the distance between them is $||uv||$
- The dark line in the picture is a long obstacle which is just above the points u and v, and is extended from u to v.
- Right on the top of the obstacle, all n sensor nodes have been placed equidistant from one another, and the distance between them is $2b$, where b is some positive value.

Therefore, here the best coverage value is b, and the dotted line is the spanner path for going from u to v which is at most $k.||uv||$ in length.

Now,

$$||uv|| = 2bn \tag{5.1}$$

, where b is the *Coverage Value*

The bounded degree k-spanner path is the dotted path, where the relationship is $Spanner_{uv} = k.(shortestpath_{uv})$. Using the property of spanners, we can see that

$$Spanner_{uv} = k.(||uv||) \tag{5.2}$$

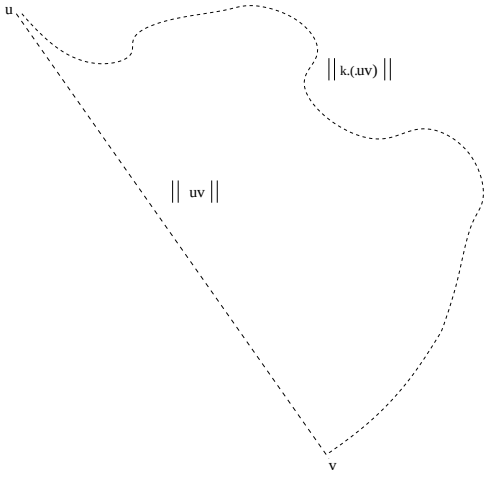

Figure 5.6. The spanner path (dotted) line is k times longer than the shortest path $||uv||$.

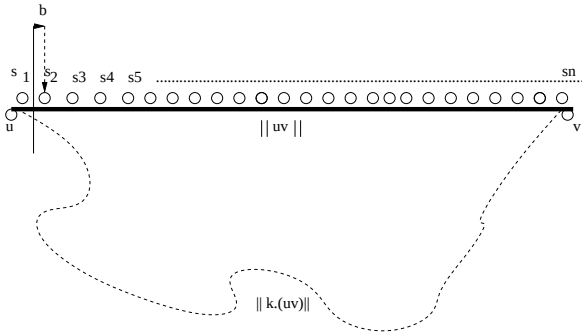

Figure 5.7. The Stretch Factor of Best Coverage of the spanner - A worst case scenario.

Substituting the value of $||uv||$ from Equation-(5.1), we see that

$$k.(||uv||) = k.2bn \Rightarrow b.(k2n) \tag{5.3}$$

thus it is proved that the approximation factor is $O(nk)$.

Next we analyze the running time of the approximation algorithm.

Lemma 3 *The running time of the Approximated $BCP(s,t)$ for transparent obstacle is $O(nm + n^2)$.*

Proof 6: The only difference between this approximation algorithm and its exact version is the difference in the constructed graph. While the visibility graph has a quadratic number of edges, the k-spanner has a linear number of edges. Similar to the

exact algorithm, the running time here is dominated by the edge weighing step. Thus the overall running time of this algorithm is $O(nm + n^2)$.

5.3 Summary

In this Chapter, we proposes two algorithms in the computation of $BCP(s, t)$ for transparent of obstacles; one exact algorithm and one approximation algorithm. In separate sections, we outline each of these two algorithms, illustrate the edge weighing policy, illustrate the algorithm. We conclude each of these cases by proving its correctness and analyzing the running time. In addition to that, we calculate the approximation factor of our proposed approximation algorithm.

CHAPTER 6

CONCLUSION

In this work, we model the presence of line segment obstacles and its impact in the computation of a exposure-based path in Wireless Sensor Networks. Precisely, in presence of a set of obstacles, we investigate the computation of Best Coverage Path, denoted as $BCP(s,t)$, between a source s and destination point t in the sensing field. We have shown that obstacles complicate the problem. In this work, we illustrate the idea of two different variations of obstacles - opaque and transparent obstacles. Depending on these two variants, we define two separate problems, the $BCP(s,t)$ for opaque obstacles and the $BCP(s,t)$ for transparent obstacles.

We have proposed an $O((m^2n^2+n^4)\log(mn+n^2)))$ time algorithm for computing $BCP(s,t)$, given n sensor nodes and m opaque line obstacles. The algorithm is based on constructing a special dual of the Constrained Voronoi Diagram; the time complexity of this algorithm is dominated by the quartic combinatorial structure of the diagram itself. For transparent obstacles, we design two solutions; an exact algorithm and a more time-efficient approximation algorithm. The exact $BCP(s,t)$ for transparent obstacles takes $O(nm^2+n^3)$ time for computation. This algorithm is based on constructing the visibility graph of the sensors and obstacles (with special weights assigned to edges). The approximation algorithm for computing $BCP(s,t)$ takes $O(nm+n^2)$ time. This algorithm is based on constructing a k-spanner of the visibility graph. In addition, we derive the approximation factor of the cover value of the computed path for this algorithm. We as well analytically prove the correctness of our proposed solutions. These constitute the main results of this thesis in brief.

As future scope of this research work, we intend to extend these ideas for modeling more complex obstacles, such as polygonal obstacles. As well, we like to investigate the

best coverage path in presence of polygonal/line segment obstacles for heterogeneous sensor network. In addition to that, we plan to investigate practical techniques such as heuristics and randomized algorithms for solving these problems efficiently which can reduce the running time. A nice improvement of this work can be to construct/pre-process the constrained Voronoi diagram and then design an incremental algorithm as queries at run time to compute the best coverage path. Furthermore, the *Worst Coverage Path* problem *amidst obstacles* is an interesting problem for future research.

BIBLIOGRAPHY

[1] A. Baltsan and M. Sharir, *"On shortest paths between two convex polyhedra."* J. ACM 35, 267-287, 1988.

[2] A. Howard, M. J. Matari, and G. S. Sukhatme. *"An incremental self-deployment algorithm for mobile sensor networks." Autonomous Robots Special Issue on Intelligent Embedded Systems*, 13(2):113-126, 2002.

[3] A. Howard, M. Mataric and G. Sukhatme, *" Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem, "* In Proceedings of the 6th International Symposium on Distributed Autonomous Robotic Systems - DARS02, pages 299-308, Fukuoka, Japan, June, 2002.

[4] A. Lingas. *"Voronoi diagrams with barriers and their applications." Inform. Process. Letters*, 32:191–198, 1989.

[5] Amitabha Ghosh, Sajal K. Das, *"Distributed Greedy Algorithm for Connected Sensor Cover in Dense Sensor Networks"*,International Conference on Distributed Computing in Sensor Systems, June 2005.

[6] Amitabha Ghosh and Sajal K. Das, *" Coverage and Connectivity Issues inWireless Sensor Networks"*. Book Chapter.

[7] B. Aronov. *"On the geodesic Voronoi diagram of point sites in a simple polygon."* Algorithmic 4, 109140, 1989.

[8] B. Liu and D. Towsley. *" On the coverage and detectability of wireless sensor network." Proc. of WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, 2003.

[9] C. A. Wang and Y. H. Tsin. *"Finding constrained and weighted voronoi diagrams in the plane." Computational Geometry: Theory and Applications*, 10(2):89 – 104, MAY 1998.

[10] C. A. Wang and L. Schubert, *" An optimal algorithm for constructing the Delaunay triangulation of a set of line segments."* Proc. 3rd Ann . ACM Symp. Computational Geometry, 223 - 232, 1987 .

[11] C. A. Wang, *" A new generalization of Voronoi diagrams in the plane."* Ph .D . Thesis, Univ . of Alberta, Canada, 1987 .

[12] C. Burnikel, K. Mehlhorn, and S. Schirra. *" How to compute the Voronoi diagram of line segments: Theoretical and experimental results."* manuscript.

[13] C. S. Meguerdichian, F. Koushanfar, M. Potkonjak, and M. Srivastava. *"Coverage problems in wireless ad-hoc sensor networks." Infocom.*, APRIL 2001.

[14] D. Tian and N. D. Georganas, *"A CoveragePreserving Node Scheduling Scheme for Large Wireless Sensor Networks."* Proc. of the 1st ACM Workshop on Wireless Sensor Networks and Applications, 2002.

[15] D. W. Gage. *"Command control for many-robot systems." Nineteenth Annual AUVS Technical Symposium, Reprinted in Unmanned Systems Magazine.*, 10(4):28–34, JANUARY 1992.

[16] F. Aurenhammer and G. Stckl. *" On the peeper's voronoi diagram." SIGACT News.*, 22(4):50–59, 1991.

[17] F. Aurenhammer. *Voronoi diagrams: A survey of a fundamental geometric data structure.* ACM Comput. Surv., 23(3):345-405, Sept. 1991.

[18] G. Das. *" The visibility graph contains a bounded-degree spanner." Proc. 9th Canad. Conf. Computational Geometry*,1997.

[19] G. Das. *" Approximation Schemes in Computational Geometry"*. Ph.D Thesis, Univ . of Wisconsin-Madison, 1990 .

[20] G. S. Sukhatme and M. J. Mataric, *" Embedding Robots into the Internet."* Comm. ACM, vol. 43, pp. 67–73, May 2000.

[21] G. Veltri, Q. Huang, G. Qu, and M. Potkonjak. *"Minimal and maximal exposure path algorithms for wireless embedded sensor networks. "* In Proceedings of the 1st

International Conference on Embedded Networked Sensor Systems - Sensys03, pp. 40-50, Los Angeles, CA, Nov 2003.

[22] H. Gupta, S. R. Das, Q. Gu," *Connected Sensor Cover: Self-Organization of Sensor Networks for Efficient Query Execution ",* MobiHoc, June 2003.

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

[24] J. Hershberger and S. Suri, *" Efficient Computation of Euclidean Shortest Path in the Plane. "* In Proc. of 34th Annual IEEE Sympos. Found. Comput. Sci. , 508–517, 1993.

[25] J. ORourke. *"Art gallery theorems and algorithms."* Oxford University Press, Inc., Oxford, 1987.

[26] J. S. B. Mitchell, *" Shortest Paths Among Obstacles in the Plane."*, In Proc. 9th Annual ACM Sympos. Computational Geometry, 308–317, 1993.

[27] J. Urrutia. *" Art gallery and illumination problems." In Handbook on Computational Geometry, Elsevier Science Publishers, J.R. Sack and J. Urrutia, eds*, 973–1026, 2000.

[28] K. J. Supowit. *The Relative Neighborhood Graph With An Application To Minimum Spanning Tree.*, Journal of Associate Computing Machine, no 30, 1983.

[29] K. Clarkson. *" Approximation algorithms for shortest path motion planning." Proc. of the nineteenth annual ACM conf. on Theory of computing*, 56-65, 1987.

[30] L. P. Chew. *"Constrained delaunay triangulations." Proceedings of the third annual symposium on Computational geometry*, 215–222, 1987.

[31] L. P. Chew,*"Constrained Delaunay triangulations ."* Algorithmica 4, 97 -108, 1989 .

[32] L. P. Chew, *" There are planar graphs as good as the complete graph. "* Journal of Computer and System Sciences, 39, 205–219, 1989.

[33] M. Berg, M. Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer, 1997.

[34] M. Cardei, D. MacCallum, X. Cheng, M. Min, X. Jia, D. Li, and D. Z. Du, *"Wireless Sensor Networks with Energy Efficient Organization."*, Journal of Interconnection Networks, Vol 3, No 34, pp 213-229, Dec 2002.

[35] O. Khatib. *"Real-time obstacle avoidance for manipulators and mobile robots."* International Journal of Robotics Research, 5(1): pp. 90-98, 1986.

[36] R. Seidel. *"Constrained delaunay triangulations and voronoi diagrams with obstacles." Rep. 260, IIG-TU Graz, Austria*, 178–191, 1988.

[37] S. Ghosh and D. Mount. *" An output sensitive algorithm for computing visibility graphs." SIAM Journal of Computing,*, 20(5):888–910, 1991.

[38] S. Megerian, F. Koshanfar, M. Potonjak, and M. B. Srivastava. *"Worst and best-case coverage in sensor networks." IEEE Transaction for Mobile Computing*, 4:84–92, 2005.

[39] S. Meguerdichian, F. Koushanfar, G. Qu, and M. Potkonjak. *" Exposure in wireless ad hoc sensor networks." Procs. of 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)*, 139–150, JULY 2001.

[40] S. Poduri and G. S. Sukhatme. *"Constrained coverage in mobile sensor networks."* In Proceedings of IEEE International Conference on Robotics and Automation - ICRA04, pp. 40-50, New Orleans, LA, AprMay 2004.

[41] S. S. Dhillon and K. Chakrabarty, *" Sensor placement for effective coverage and surveillance in distributed sensor networks"*, Proc. of IEEE Wireless Communications and Networking Conference, vol. 38, no. 9, pp. 1609-1614, 2003.

[42] S. Slijepcevic and M. Potkonjak,*" Power Efficient Organization of Wireless Sensor Networks"*, ICC. Helsinki, pp. 472-6, June 2001.

[43] T. Asano, L. J. Guibas, J. Hershberger, and H. Imai, *" Visibility of Disjoint Polygons"*. Algorithmica , 1: 49–63, 1986.

[44] T. J. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press and McGraw-Hill, 1990.

[45] Tsuy Uki Okabe, Barry Boots and Kokichi Sugihara *" Spatial Tessellations-Concepts and Applications of Voronoi Diagrams."* Chichester:John Wiley and Sons, 1992.

[46] X. Y. Li, P.-J. Wan, and O. Frieder. *" Coverage problems in wireless ad-hoc sensor networks." IEEE Transactions for Computers*, 52:753-763, JUNE 2003.

[47] Xiang Yang Li, Peng-Jun Wan, Wang Yu. *" Power Efficient and Sparse Spanner for Wireless Ad Hoc Networks."* IEEE International Conference on Computer Communications and Networks (ICCCN01), Scottsdale, Arizona, Oct 15–17, 2001.

[48] Xing, Wang, Zhang, Lu, Pless, and Gill. *" Integrated coverage and connectivity configuration for energy conservation in sensor networks"*, TOSN 1(1), pp. 36-72, 2005.

[49] Y. Zou and K. Chakrabarty. *"Sensor deployment and target localization based on virtual forces"*. In Proceedings of IEEE Infocom - INFOCOM 03, pages 1293-1303, San Francisco, CA, Apr 2003.

[50] Y. Zou and K. Chakrabarty. *"Sensor deployment and target localization in distributed sensor networks"*. Transactions on IEEE Embedded Computing Systems, 3(1): 61-91, 2004.

[51] Z. Zhou, S. Das, H. Gupta,*"Fault Tolerant Connected Sensor Cover with Variable Sensing and Transmission"*,IEEE SECON, 2005.

[52] Zongheng Zhou, Samir Das and Himanshu Gupta.*"Variable Radii Connected Sensor Cover in Sensor Networks"*, A Report. Department of Computer science, Stony Brook University.

BIOGRAPHICAL INFORMATION

Senjuti Basu Roy received her Bachelor of Technology degree in Computer Science and Engineering from University Of Calcutta, Calcutta, India, in 2004. She began her graduate studies in the Department of Computer Science and Engineering at The University of Texas at Arlington in Fall 2005. She is expected to receive her Master of Science degree in Computer Science and Engineering from The University of Texas at Arlington in Spring 2007. Her research interest includes Computational Geometry, Sensor Network, Approximation algorithm, Randomized algorithm, online algorithm etc.